UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE APOYO A LA DOCENCIA Y GESTIÓN DE ALUMNOS EN CURSOS CON METODOLOGÍAS GRUPALES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN CIENCIAS DE LA COMPUTACIÓN

CRISTIÁN ANDRÉS SALAZAR DE LA FUENTE

PROFESOR GUÍA:
MATÍAS TORO IPINZA

MIEMBROS DE LA COMISIÓN:
JOSE PINO URTUBIA
LUIS MATEU BRULÉ

SANTIAGO DE CHILE
2024

Resumen

La implementación de metodologías grupales en la educación superior ofrece una serie de ventajas, como un mayor compromiso y habilidades sociales entre los estudiantes, así como una comprensión más profunda de los contenidos académicos. Sin embargo, gestionar estos cursos presenta desafíos únicos tanto para docentes como para alumnos, incluyendo la coordinación de actividades, la gestión de equipos, la evaluación de trabajos grupales y la retroalimentación oportuna. Estos aspectos críticos implican una carga adicional para los equipos docentes, que deben asegurar el traspaso de conocimientos, resolver dudas, corregir entregas y motivar a los estudiantes para seguir aprendiendo.

Esto se ve reflejado en las dificultades que están presentando los equipos docentes y estudiantes de cursos masivos de computación, como Bases de Datos del Departamento de Ciencias de la Computación de la Universidad de Chile, para manejar de forma ordenada la administración y seguimiento de equipos y sus evaluaciones.

Con este problema en mente, y con el objetivo de mejorar el proceso actual que se realiza en dichos curso, en la presente memoria se detalla sobre el diseño e implementación de una aplicación web y móvil para el apoyo a la docencia y la gestión de alumnos en cursos masivos de computación con metodologías grupales.

Para la implementación se utilizó una metodología SCRUM, planificando las tareas tareas que entregaran mayor valor. Esta aplicación permite el ingreso de los usuarios, visualizar sus cursos con sus roles, administrar y crear grupos y equipos; ver, crear y administrar tareas y la recepción de entregas a las tareas, además de que cada entrega pueda tener un estado y recibir comentarios de los docentes. Las herramientas utilizadas para el desarrollo de esta aplicación fueron Django REST Framework y React Native con Expo.

Para la validación de la solución se realizó una Encuesta de Usabilidad con 18 estudiantes del curso Bases de Datos, quienes calificaron la aplicación en una escala del 1 al 5 sobre 10 distintos aspectos. Esta evaluación tuvo un puntaje positivo y, por lo tanto, se califica como una aplicación de usabilidad aceptable.

Para futuros trabajos se propone la implementación de más funcionalidades que permitan el uso consolidado de la aplicación en los cursos masivos, como evaluación con nota, exportar datos a U-Cursos, notificaciones y aplicar cambios sugeridos en la validación.

"Un buen profesor es como una vela. Se consume iluminando el camino de otros."

Agradecimientos

Quiero agradecer la suerte que tuve de cruzarme con esta carrera. Me daba vergüenza decir que considero que es una "bonita carrera" (sobre todo cuando pienso en que quise estudiar actuación en vez de esto), pero ahora lo digo con felicidad y seguridad.

Quiero agradecer a esta facultad que me recibió tan llena de profesores, estudiantes, pasillos, polvo, libros, computadores, sensores, ventiladores, ecos, lugares de estudio, puestas de sol, dolores de cabeza, amores, carretes, funcionarias y funcionarios (que siempre me indicaron que estaba parado en la oficina incorrecta), pocos lugares para llorar, amiges, sillones, luces led, enemigos, arboles imposibles, canchas, cables, tizas, pasa-diapositivas, micrófonos, cócteles, comida callejera, almuerzo con manchas de pasto en la ropa, sushi con escherichia coli, torniquetes, tótems, que fueron elementos que solo sumaron a mi experiencia por la Universidad y que me permitieron desear y quererla.

Quiero agradecer a mi mamá y a mi papá (o a mi papá y a mi mamá), por ser como son, por obligarme a terminar la carrera (con todo el cariño del mundo), por tenerme comida al llegar, por preguntarme como estoy, por quererme y confiar en mi. Agradezco también a mi hermana por pedirme aparecer aquí.

Me gustaría agradecer también a mis compañeros de Teatro Beauchef cumplir el deseo de ser otra persona durante una hora y media a la semana, en una recóndita sala en la facultad; a mis compañeres de Beauchef Diverso, por enseñarme a ser un gay funcional, orgulloso y feliz; al CaDCC por demostrarme que somos adictos al karaoke, las orejas de gato y los jueguitos de pc; a la Biblioteca Central, por demostrarme que cultura e ingeniería pueden llevarse muy bien y a la DDG, por confiar en los estudiantes cuando se trataba de realizar cambios profundos sobre el género y la diversidad en nuestra facultad.

Por último, quiero agradecer a personas en particular: Melanie Marquez, por los micro-perreos, Alejandro Rivas, por ser mi primer amor de universidad, Majo Ganora, por ser mi reencarnación oficial en la facultad y a Leonardo Falcón, por ayudarme a sublimar mis frustraciones artísticas.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

La implementación de metodologías grupales en la educación superior ha demostrado generar múltiples beneficios, incluyendo un mayor compromiso de los estudiantes, un desarrollo de habilidades sociales y de trabajo en equipo, así como una comprensión más profunda de los contenidos académicos. Sin embargo, la gestión de estos cursos presenta desafíos únicos tanto para los docentes como para los alumnos. La coordinación de actividades, la gestión y registro de equipos, la evaluación de entregas grupales y la retroalimentación oportuna son aspectos críticos que requieren atención constante.

Y estos aspectos críticos implican una labor de gestión, que lleva a una carga adicional implícita para los equipos docentes de cada uno de los cursos, los cuales tienen como objetivo explícito la traspaso de conocimientos, resolución de dudas, corrección de entregas e incentivar y motivar a que el estudiantado siga aprendiendo.

Actualmente, en la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile (FCFM), en la carrera de Ingeniería Civil en Computación, existen diversos cursos del Departamento de Ciencias de la Computación (DCC) como:

- CC1002 Introducción a la Programación
- CC3201 Bases de Datos
- CC5212 Procesamiento Masivo de Datos
- CC5215 Privacidad de Datos
- CC5205 Minería de Datos
- CC7220 La Web de Datos

que poseen dinámicas de evaluación grupales. Además, dentro de estos cursos existen distintos tipos de grupos, como “Laboratorios” y “Proyectos”, donde cada uno abarca aspectos de aprendizaje distintos.

Para estos cursos, así como para los demás de la Universidad de Chile, se cuenta con el apoyo digital de la herramienta U-Cursos, que permite, la publicación de eventos, tareas, notas, entre otros, para agilizar el proceso educativo e incentivar la participación. Sin embargo, dentro de los últimos años y con la creciente demanda de los cursos en la carrera de Computación, la gestión de estos cursos se ha dificultado, reflejándose en una carga mayor para los equipos docentes, retraso en la entrega de notas y retrasos en los objetivos pedagógicos, debido a que no es utilizable para todas las necesidades que se tienen en estos momentos.

1.2. Descripción del problema

Actualmente, ocurren ciertas situaciones dentro de los cursos del DCC que entorpecen y ralentizan tanto la labor docente como el aprendizaje de los estudiantes, las cuales serán descritas a continuación.

1. Los estudiantes no saben en qué sala de las asignadas al curso se encuentran sus compañeros de equipo. Esto produce no solo pérdida de tiempo, si no que también aglomeraciones fuera de las salas.
2. Los estudiantes pierden tiempo al inicio de la instancia de trabajo buscando formar un equipo. Además, dado que los integrantes de cada equipo podrían cambiar semana a semana, este problema puede persistir durante todo el semestre.
3. Si bien U-Cursos permite la creación de grupos, esta funcionalidad no es utilizada por el equipo docente por ser considerada tediosa. Crear grupos implica realizar aproximadamente 200 clics (la cantidad de estudiantes en un curso) en una matriz de 200 x 65 aproximadamente (65 es la cantidad aproximada de equipos de 2-3 personas).
4. Como no se configura las entregas para que sean grupales, a veces, más de una persona del equipo entrega el mismo desarrollo, lo que provoca que dos ayudantes, por confusión, corrijan la misma entrega.
5. Ayudantes pierden tiempo cruzando la información de la asistencia con la de los integrantes de cada equipo. Deben revisar la sección integrantes de cada entrega, y verificar en U-Cursos que todo el grupo haya asistido a una instancia de trabajo, si esta es con asistencia obligatoria.
6. Existe un retraso en la entrega de las notas por falta de coordinación en la división de evaluaciones. Esto ocurre por la gran cantidad de ayudantes (que pueden llegar a ser ocho). También ocurre que, por des-coordinación, dos ayudantes corrigen el mismo lote de laboratorios, perdiendo bastante tiempo en un mal entendido.

Y con respecto a los proyectos, y en particular para el curso CC3201 Bases de Datos, encontramos que:

1. **Conformación de equipos**: Los estudiantes pierden tiempo al inicio del curso buscando formar un equipo. Actualmente forman equipos a través de una planilla compartida

de Google Sheets, en la que grupos incompletos y estudiantes sin grupo se ingresan con dicho estado. El equipo docente no tiene una forma de emparejar automáticamente dichos grupos y lo realizan de forma manual. Tampoco tienen una forma de notificarle a los estudiantes que se les ha asignado un grupo, retrasando el avance del proyecto.

2. **Retroalimentación del dataset**: El proceso de elección de tema/dataset para el proyecto semestral y posterior retroalimentación es confusa, donde mensajes entre equipo docente y grupos se pierde. Este proceso se ha tratado de resolver de dos formas distintas:

   (a) A través del foro de U-Cursos, el cual es desordenado porque en un mismo hilo conversan simultáneamente varios equipos, por lo tanto, a todos los estudiantes les llega una notificación por la revisión de otros grupos, y deben revisar constantemente si su dataset ha sido aprobado, rechazado y/o retroalimentado. Además, en el mismo mensaje, una persona del grupo debe reportar a los integrantes del grupo, lo que recarga aun más la labor del equipo docente, teniendo que ingresar dicha agrupación a mano a U-Cursos.

   (b) Utilizando Google Sheets, donde se colocan los integrantes en las primeras columnas, y el dataset del equipo en otra. En esta modalidad, cuando un equipo recibe una retroalimentación, no reciben ningún tipo de notificación, por lo que el *feedback* no es fluido. Además, cuando cada grupo recibe una retroalimentación, el equipo docente cambia el valor de una celda, que indica por ejemplo, que fue rechazada su propuesta de base de datos, y deben cambiarla. Esto no es notificado a los estudiantes, los cuales no están constantemente revisando la planilla. Cuando la cambian, a veces los alumnos olvidan cambiar el estado de la retroalimentación, lo que retrasa todo el proceso, ya que el equipo docente no sabe que tiene que volver a revisar su caso.

A partir de lo anterior, se plantea el siguiente problema: ¿Como se podría apoyar a la docencia y gestión de alumnos en estos cursos con metodologías grupales para solventar estos problemas?

1.3. Objetivos

A continuación, se presentará el objetivo general de esta memoria junto con los objetivos específicos a cumplir.

1.3.1. Objetivo general

Implementar una aplicación multiplataforma (móvil y web) que permita asistir y apoyar a docentes y estudiantes en la gestión de cursos masivos con metodologías grupales.

1.3.2. Objetivos específicos

1. Reemplazar el flujo actual de conformación de equipos de un curso.
2. Implementar la recepción de entregas de las tareas en formato PDF o archivo de texto por parte de un equipo.
3. Integrar a la aplicación el registro de la asistencia desde U-Cursos para relacionar entregas con asistencia.
4. Permitir la distribución de equipos de estudiantes a las salas de trabajo.
5. Implementar la asignación automática de ayudantes para corregir cada entrega.
6. Implementar notificaciones que permitan dar aviso de cambios en la aplicación que requieran una acción del usuario.

1.4. Evaluación

A continuación se describirá la forma de evaluación del trabajo.

El éxito de la solución a desarrollar se verá demostrado en la mitad del proceso de desarrollo, probando un MVP en el curso CC3201. Al finalizar este desarrollo se evaluará la usabilidad y la utilidad del nuevo sistema con un grupo de estudio de estudiantes, auxiliares y ayudantes que hayan pasado por alguno de los cursos mencionados. Tanto la usabilidad como la utilidad de la herramienta se evaluarán haciendo uso de instrumentos ya definidos, como lo es la Encuesta de Usabilidad (System Usability Scale).

1.5. Solución propuesta

En esta sección se realizará una descripción general de la solución propuesta.

Para lograr el objetivo planteado se desarrollará una aplicación que permita la concentración y administración de los datos sobre los cursos, sus integrantes, sus grupos, salas asignadas, laboratorios y proyectos.

La solución contempla una aplicación móvil y una aplicación web:

- **Aplicación móvil**: Se espera que los usuarios requieran de revisar información antes de sentarse dentro de la sala con sus computadores, por lo tanto, una aplicación móvil les permitirá, de forma cómoda, consultar información sobre la sala asignada, grupos, de forma cómoda.
- **Aplicación web**: Se espera que estudiantes quieran subir su desarrollo de laboratorio y proyecto a la aplicación. El desarrollo siempre lo realizan desde un computador. Además se espera que las entregas sean descargada por ayudantes para su corrección, también desde un computador.

Ambas aplicaciones compartirán interfaz, pero adaptada a distintos dispositivos (en tamaño y orientación).

1.5.1. Módulos

La aplicación tendrá 4 tipos de usuarios: Docente, Auxiliar, Ayudante y Estudiante, que corresponden a los roles que actualmente tienen la mayoría de los cursos ofrecidos por el Departamento de Ciencias de la Computación de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Además, se mantendrá información sobre el curso, su semestre, y los grupos formados para cada laboratorio y para el proyecto. Usuarios serán cargados al principio de cada semestre y se autentificarán solamente con su rut (como nombre de usuario y contraseña).

La implementación de la aplicación se dividirá en módulos, que permitirán separar cada funcionalidad.

- **Autenticación**: corresponde al registro, carga e ingreso de usuarios.
- **Grupos**: Concentra las funcionalidades de creación y administración de tipos de grupos y equipos dentro de un curso.
- **Entregas**: corresponde al manejo de entregas (archivos) a las tareas que posee cada curso. Además, corresponde a la asignación de entregas al equipo docente para su corrección.
- **Salas**: permitirá administrar y manejar carga balanceada de estudiantes, grupos y equipo docente en cada una de las salas.
- **Notificaciones**: correo o notificaciones *push* a pedir a cada usuario que ejecute una acción, como recordatorio, cuando se estime conveniente.

1.6. Metodología

Durante el desarrollo de esta aplicación se adoptó una metodología SCRUM, priorizando las tareas de forma iterativa, pensando en los módulos a implementar de forma incremental.

En esta sección se listarán las diferentes tareas a seguir para el desarrollo de la solución propuesta, en el orden de prioridad propuesto por el memorista.

1. Implementar modelo de datos
2. Implementar módulo de grupos
3. Implementar módulo de entregas
4. Montar aplicación

5. Ejecutar pruebas en curso real
6. Implementar módulo de notificaciones
7. Implementar módulo de salas
8. Ajustes de sistema a partir de pruebas
9. Escribir la memoria

Se eligió este orden de tareas ya que se desea probar la funcionalidad de grupos y proyectos a la mitad del segundo semestre del 2023, en un curso real. Además, este orden permite implementar primero lo más importante para obtener un producto que ponga en marcha un nuevo flujo de conformación de equipos y entrega de tareas.

1.7. Estructura del informe

El presente informe se dividirá en 6 capítulos:

1. **Introducción**: primero se describe y detalla el contexto del presente trabajo, una descripción sobre el problema en general y su solución propuesta a grandes rasgos. Luego, se plantean el objetivo general y los objetivos específicos de este trabajo. Por último, se describe la metodología a seguir para el desarrollo de esta memoria.
2. **Antecedentes**: en este capítulo se discuten las distintas soluciones existentes o competencias directas a la nueva propuesta de solución. Además se discuten las distintas tecnologías disponibles para el desarrollo de aplicaciones móviles y web.
3. **Análisis y diseño**: se detallan los requerimientos funcionales necesarios para el desarrollo de la aplicación, así como las decisiones de diseño tomadas para la misma. Se presenta además el diseño del flujo deseado en la aplicación, junto con el diseño de las interfaces y la arquitectura completa de la aplicación.
4. **Implementación**: se describe, de forma exhaustiva, el proceso de desarrollo e implementación de la solución propuesta. En esta se abarcan aspectos del backend, frontend y la implantación del sistema.
5. **Validación**: Se presenta el diseño experimental realizado para validar la usabilidad de la aplicación, junto con la exposición de los resultados de esta y un análisis sobre los resultados obtenidos.
6. **Conclusión**: se realiza una revisión completa sobre el cumplimiento de los objetivos y un análisis del alcance de la solución. Además, se realiza una reflexión realizado considerando varios aspectos del desarrollo. Finalmente, se exponen posibles oportunidades de desarrollo a futuro de el presente trabajo.

Capítulo 2

Antecedentes

2.1. Herramientas utilizadas actualmente

A continuación, se discutirán las soluciones existentes relacionados con el problema planteado. Actualmente en el mercado existen varias alternativas a aplicaciones que permiten manejar y gestionar cursos, junto con las tecnologías que actualmente se utilizan para los cursos planteados en el capitulo anterior:

U-Cursos

U-Cursos es una plataforma de apoyo al desarrollo de la docencia y de los procesos de enseñanza-aprendizaje, basada completamente en tecnología Internet [7]. Esta aplicación posee tanto versión de escritorio como versión móvil. Actualmente, tanto los cursos mencionados anteriormente como el resto de los cursos de la Universidad de Chile, son administrados a través de esta aplicación. En particular, funcionalidades como el registro de asistencia, elección de base de datos para el proyecto y entrega de tareas se realiza a través de sus funcionalidades.

Sin embargo, para algunas de las necesidades del curso se ha tenido que adaptar a las limitaciones que posee esta aplicación:

- Para la elección de datasets y retroalimentación se utiliza el foro, en el cual el equipo docente le responde a cada mensaje por cada propuesta de cada grupo. Esto hace que se pierdan algunos mensajes, que estudiantes reciban notificaciones de retroalimentaciones de otros grupos y no el suyo, provocando que tengan que estar todo el tiempo atentos a las respuestas o que se tenga que abrir otro foro por límite de mensajes.
- Para la entrega de laboratorios solo un integrante debe enviar su desarrollo. Si más de un integrante lo manda, puede ocurrir que dos ayudantes corrijan el mismo laboratorio, perdiendo tiempo en corrección innecesaria.

- Para la conformación de grupos y la característica de entrega por equipos, U-Cursos permite crear grupos, que es útil para las entregas de los proyectos y laboratorios, pero la creación de estos grupos se debe hacer a mano, seleccionando un *radio button* en una matriz del tamaño de la cantidad de alumnos por la cantidad de grupos (aproximadamente 150 x 50), y esto toma bastante tiempo porque se necesita ingresar un texto a cada nombre de grupo y hacer un *click* por alumno. Esta solución no escala de buena manera para cursos con más de 200 alumnos. Además, la creación de estos grupos lo debe realizar una misma persona.
- La forma de conectar la asistencia grupal con la nota de un laboratorio no es escalable por el punto anterior.
- Para la distribución de grupos en cada sala, el equipo docente no tiene una forma para avisar qué grupos van a qué sala.

Esta aplicación ha estado en constante desarrollo, mejorando año tras año con nuevas funcionalidades, pero aun así, es difícil pedir una extensión de la aplicación al equipo de desarrollo de U-Cursos, sobre todo porque dicha aplicación es utilizada también por otras universidades. Por estas razones es que U-Cursos no esta funcionando correctamente para las necesidades del curso, y a medida que aumente el número de estudiantes, los problemas sólo se agravarán.

WebCursos

WebCursos [9] es una aplicación desarrollada por Moodle, que en Chile es utilizada por la Universidad Adolfo Ibáñez como plataforma institucional para acceder al contenido digital de las asignaturas de pregrado y postgrado.

Cada curso presenta "actividades", funcionalidades que tiene disponible el equipo docente para un curso, y entre ellas está la auto-selección de grupo: permite crear y seleccionar grupos a los propios estudiantes:

1. Los estudiantes pueden crear grupos, con una descripción y protegerlos con una contraseña.
2. Los estudiantes pueden elegir y unirse a grupos. El equipo docente se puede asignar a grupos.
3. Soporta entrega de tareas por grupo.

Si bien es una funcionalidad útil, esta aplicación es contratada por universidades para tener acceso a todas las funcionalidades, por lo que su costo es muy elevado en comparación con las funcionalidades que podrían llegar a utilizarse. Además, en la Universidad de Chile, no se espera incluir aplicación para manejar cursos, solo para ser utilizado en algunos cursos con la dinámica descrita al principio de este apartado. Para este curso se requieren solo una de todas las funcionalidades que ofrece WebCursos, por lo tanto, no tendría sentido contratar un servicio grande para utilizar sólo una de las funcionalidades.

Google Sheets

Google Sheets [8] es una aplicación web de Google que permite la creación de hojas de cálculo colaborativas. En el curso CC3201, se utiliza una planilla en Google Sheets para la conformación de equipos, coordinación de los temas de proyectos, aprobar, rechazar y dar feedback a las bases de datos elegidas por los grupos:

- La conformación de equipos es lenta. Grupos pueden colocar que están buscando a un integrante o una persona puede colocar que busca grupo, y nadie se encarga de revisar y emparejar activamente a estos dos grupos debido a que no le llegan notificaciones ni a los docentes para realizar el emparejamiento, ni a los estudiantes cuando han sido asignados a un grupo. Por lo tanto, todos los integrantes del curso deben estar constantemente atentos a la planilla.
- Una vez que los grupos están formados, deben proponer una base de datos en una de las columnas y justificar su elección. En ese momento los grupos y el equipo docente se comunican a través de la columna "Estado", de forma manual, cambiando al estado entre "Aprobado", "Esperando revisión" y "Rechazado". Dado que no existen forma de notificar a cada grupo o equipo docente el cambio de un estado en la planilla, se retrasa la revisión y comunicación por ambas partes.

Esta es una de las soluciones que se utilizan actualmente y no está funcionando de forma ágil. El equipo docente está entregando una retroalimentación tardía, ya que no se les notifica que se les ha cambiado el estado. Además, mientras más estudiantes ingresan al curso, más difícil será de manejar la cantidad de grupos. Además, ninguna de ellas presenta todas las funcionalidades requeridas por los problemas planteados, o actualmente no son una buena solución ni escalable para los cursos. Es más, el uso en conjunto de estas aplicaciones no está llevando a una buena solución, ya que se desea tener toda la información en un mismo lugar.

2.2. Tecnologías para el desarrollo de aplicaciones web y móviles

Frontend

En relación al contexto desarrollado en este informe, una aplicación movil y web que permita reunir todas estas funcionalidades, podría resolver el problema planteado. Para el desarrollo de una aplicación web y móvil se desea utilizar una tecnología que tenga compatibilidad con cualquier dispositivo (Android, IOS y Web). Con esto se refiere a una forma de tener un mismo código para diferentes tipos de pantalla y sistemas operativos. Para estos efectos existen tecnologías como Flutter, React Native, Expo e Ionic, que permiten la programación de lógicas de aplicación, maneja las interacciones del usuario, donde un mismo código que compila para distintos dispositivos.

- **Flutter**: es un *framework* de código abierto desarrollado por Google para el desarrollo de aplicaciones multiplataforma (web, desktop, iOS y Android) desde un mismo código base. Su programación se realiza en Dart. Las aplicaciones desarrolladas con Flutter tendrán la misma apariencia en todos los dispositivos, sin importar los componentes nativos que tenga en el caso de los dispositivos móviles.
- **React Native y Expo**: es un *framework* de código abierto creado por Meta Platforms, Inc [4]. y se utiliza para desarrollar aplicaciones móviles (Android e iOS), permitiéndole a los desarrolladores usar React con las características nativas de dichos dispositivos. Esta tecnología utiliza Javascript (TypeScript) como lenguaje de programación. Además, cuando se utiliza junto con Expo [1], un paquete NPM que proporciona una capa de abstracción por sobre React Native para simplificar el desarrollo de aplicaciones, se pueden crear aplicaciones también para la web, utilizando el mismo código fuente que el utilizado para la versión móvil
- **Ionic** [3]: Ionic es un *framework* de desarrollo de aplicaciones móviles de código abierto. Permite crear aplicaciones multiplataforma como iOS, Android y la web, utilizando tecnologías web estándar como HTML, CSS y JavaScript. Ionic utiliza Angular como su *framework* de JavaScript para construir aplicaciones de alta calidad con una interfaz de usuario atractiva y rendimiento nativo.

Dado que se desea la implementación de una aplicación web y móvil, se utilizará React Native como *framework* para el desarrollo del *frontend* de la aplicación, puesto que existe una comunidad de desarrollo amplia y su desarrollo se realiza en los lenguajes JavaScript y TypeScript, populares en el ámbito del desarrollo de aplicaciones móviles y web. Además, al agregar el uso de Expo, se logrará simplificar el desarrollo de aplicaciones multiplataforma, ya que permite el desarrollo simultáneo de aplicaciones móviles y web haciendo uso de un mismo código fuente. Además, el memorista posee experiencia previa utilizando dichas tecnologías.

Backend

Para la gestión de la lógica de la aplicación se necesita un *backend* que permita gestionar los datos de los futuros usuarios.

Como alternativas para una aplicación se tienen las siguientes tecnologías que podrían ser compatibles con el *frontend* seleccionado anteriormente:

- **Express**: Es una infraestructura de aplicaciones web en Node.js, que fue diseñado para simplificar el proceso de creación de aplicaciones web y APIs en Node.js. Se caracteriza por ser minimalista, lo que permite desarrollar aplicaciones de manera simple y rápida. Tiene sistemas de enrutamiento para asociar solicitudes HTTP (como GET y POST) con funciones del controlador específicas, que permitan modificar los datos. Proporciona una amplia gama de middleware integrado, que permite la integración de autenticación, validación de datos, registro de solicitudes, entre otras funciones de suma importancia para una aplicación basada en usuarios. Por último, esta herramienta se integra bien con otras tecnologías como motores de bases de datos y sistemas de gestión de sesiones.

- **Django REST Framework**: Es una biblioteca de Python, que se utiliza para la implementación de APIs web utilizando el *framework* Django. A su vez, Django es un *framework* de desarrollo web de alto nivel que facilita la creación de aplicaciones robustas y seguras. Django REST Framework extiende las capacidades de Django para crear APIs de manera rápida y fácil. Una de sus caracteristicas es la serialización de datos, lo que permite convertir los modelos creados en Django a otros formatos como JSON o XML, esto para facilitar la transferencia de datos entre la aplicación y el cliente que consume la API. Además ofrece soporte integrado para varios métodos de autenticación, como autenticaciones de token o autenticación de sesión. Por último, posee controladores (también llamadas *views*) basadas en clases, que simplifican la creación de puntos de acceso a la API.

Aunque ambas tecnologías son igualmente útiles y cumplen con lo requerido para el desarrollo de el proyecto, para el *backend* de la aplicación se propone el uso del *framework* Django REST, debido a la familiaridad que tiene el memorista tanto con Python como con el *framework* Django, que permite la creación de API's Web con varias funcionalidades ya implementadas, sin tener que reescribir código ni tomar tanto tiempo en el desarrollo.

Por último, este framework, que es compatible con diversos motores de bases de datos, se elige utilizar PostgreSQL, debido a la familiaridad que el practicante tiene con la configuración de este motor de bases de datos junto con Django.

En la figura 2.1 se muestra un diagrama de la arquitectura general de la aplicación basada en las tecnologías seleccionadas para el desarrollo de esta memoria.

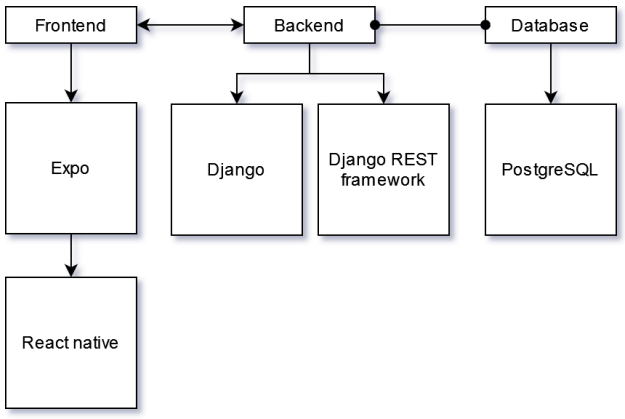


Figura 2.1: Arquitectura general de la aplicación.

Capítulo 3

Diseño de la solución

En este capítulo se explica en detalle los aspectos de diseño de la solución, junto con exponer y explicar los distintos diagramas tanto para la navegación de la aplicación, sus interfaces y el modelo de datos.

3.1. Modelo de Datos

A continuación, se explicará el modelo de datos utilizado para la estructura y lógica de la información de la aplicación.

En la figura 3.1 se pueden apreciar las distintas entidades, relaciones con sus multiplicidades y llaves primarias que posee el modelo entidad-relación. Este diagrama fue construido basandose en las convenciones seguidas en el libro de Ramakrishnan, R., & Gehrke, J. (2006). *Sistemas de gestión de bases de datos.*

A continuación se describirán en detalle cada uno de los componentes de este modelo.

3.1.1. Entidades

Dentro de las entidades en el modelo podemos encontrar las siguientes:

1. **User**: corresponde a un usuario de la aplicación. Este corresponde a una persona, donde se almacenará su información personal y se individualiza de otras instancias por su rut, que es su llave primaria.
2. **Role**: corresponde a los roles posibles de un usuario dentro de un curso. Estos son cuatro: Estudiante, Docente, Auxiliar y Ayudante.
3. **Course**: corresponde al curso. Este se identifica con su código único, año dictado, semestre dictado y sección. Además tiene un atributo de nombre de curso y un valor booleano que indica si este curso sigue activo o ya ha finalizado.

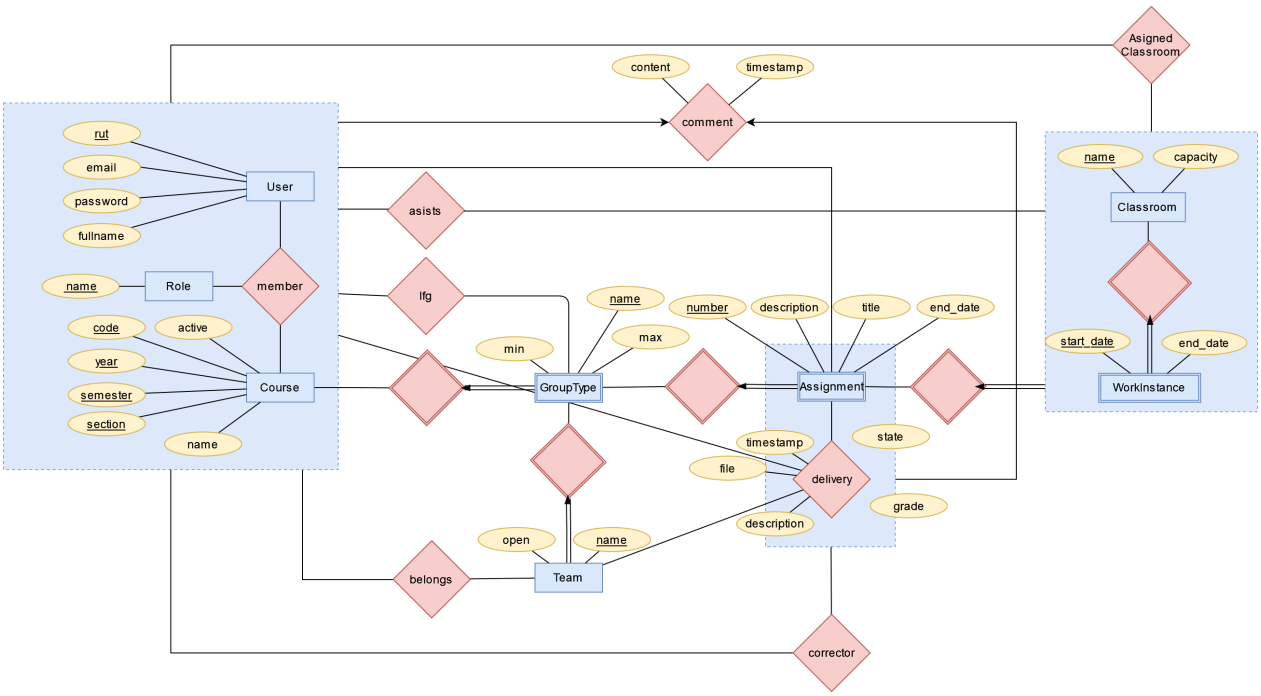


Figura 3.1: Modelo entidad-relación

4. **Classroom**: contiene información sobre el nombre de la sala y su capacidad. Este último atributo es importante ya que se necesita repartir a varios usuarios en salas y se necesita contemplar su capacidad.

5. **Team**: corresponde a un grupo de trabajo para algún tipo de grupo. Cada grupo se identifica con su identificador único. Tiene un nombre y el atributo *open* se indica si está abierto a recibir más integrantes o no. Además, el atributo *state* permite saber si el equipo está incompleto, lleno, o tiene suficiente cantidad de integrantes según el tipo de grupo.

3.1.2. Entidades débiles

A continuación se describen las entidades débiles presentes en el modelo entidad-relación. Estas entidades débiles fueron agregadas como tales debido a que estas no pueden representarse a si mismas de forma independiente. Estas entidades dependen de la existencia de otras entidades para identificarse.

1. **GroupType**: este es un tipo de grupo que tiene un curso. Estos tipos de grupos pueden ser de Control, Laboratorio, Proyecto, o cualquier otro que se requiera. Cada tipo de grupo tiene restricciones de mínimo y máximo de integrantes. Si el curso no existiera, este tipo de grupo tampoco lo haría. Además un tipo de grupo existe solo en el contexto de un curso.

2. **Assignment**: modela una tarea que requiere una entrega por parte de los estudiantes. Entre ellos están por ejemplo: Laboratorio 1, Entrega parcial 2, entre otros. Esta tarea depende del tipo de grupo al que corresponde.

3. **WorkInstance**: es una instancia de trabajo, que ocurre entre dos marcas de tiempo (hora y fecha). Además esta se identifica con la sala en la que ocurre esta instancia de trabajo y la tarea (Assignment) en la que se debe trabajar en ese tiempo. Sin un *Assignment* en el que trabajar, esta *WorkInstance* no existiría

3.1.3. Entidades virtuales

Además, se han agregados entidades virtuales, que corresponde a una asociación entre conjuntos de entidades relacionadas. Se realiza esta modelación debido a que posteriormente se requiere relacionar el conjunto de entidades relacionadas con otras entidades.

1. **Member**: es un miembro de un curso. Este tiene las llaves primarias de un usuario, un curso y su rol en el curso.
2. **Delivery**: corresponde a una entrega realizada por un equipo para una de las tareas. Esta tiene los atributos de marca de tiempo de la entrega, el estado de la entrega, una descripción que corresponde a texto entregado por el grupo para dar alguna observación, el archivo de entrega y la nota que tiene esta entrega.

3.1.4. Relaciones

A continuación, se describirán las relaciones presentes en el modelo, las cuales permiten relacionar dos o más relaciones

1. **Belongs**: relaciona las entidades Team y Member para indicar qué miembros están juntos en un grupo.
2. **Comment**: almacena uno o varios comentarios junto con su marca de tiempo realizado por un miembro a una entrega.
3. **Corrector**: corresponde a la relación que indica que miembro debe corregir qué entregas.
4. **AssignedClassroom**: relaciona una instancia de trabajo con un miembro. Esta relación permite modelar el lugar físico o evento al que debe ir a trabajar un miembro del curso.
5. **Assists**: relaciona a un miembro de un curso asistiendo a una instancia de trabajo.
6. **LookingForGroup**: indica que un miembro está sin grupo para un tipo de grupo de un curso y está en búsqueda de uno.

3.2. Estados

Con respecto a los estados que puede tener una entrega de alguna tarea, estos pueden tener 5 estados distintos, según lo siguiente:

1. *Sin revisión*: esta es una entrega enviada pero que no ha sido revisada o evaluada por alguna persona del equipo docente.
2. *Aprobado*: implica que dicha entrega ha sido aprobada con cierto puntaje sobre el mínimo reprobatorio (comúnmente 4.0) o que se le da el visto bueno para continuar con la siguiente tarea.
3. *Rechazado*: corresponden a las entregas que han sido reprobadas por estar bajo el puntaje mínimo, o son rechazadas como propuesta de entrega a una tarea.
4. *Reenviar*: indica que esta entrega no cumple a la totalidad con lo pedido pero se le deben aplicar modificaciones para que si lo haga.
5. *Inválida*: registra cuales entregas no son válidas de ser revisadas, por no ser la última versión de una entrega o porque un subconjunto de estudiantes del equipo envió otra entrega.

3.3. Usuarios

En la solución propuesta se contará con un mismo usuario que tendrá distintas funcionalidades en base a cada curso en el que se encuentre como miembro. Esto ya que puede ocurrir que un usuario sea estudiante en un curso, pero profesor auxiliar en otro. Por lo tanto, se necesita que los roles y privilegios se adapten a esta información.

Para la prueba de la aplicación, la información de los usuarios contemplará un correo electrónico, su nombre completo y RUT. Por simplicidad, para la autenticación en la aplicación de cada usuario, se utilizará como usuario y contraseña su número de RUT.

A continuación, se describirán los roles que podrá tener un usuario en el contexto de un curso, que están basados en los roles que tienen los cursos ofrecidos por el DCC:

1. Estudiante: corresponden a los usuarios que están cursando estudios dentro de uno o más cursos. Estos son evaluados a través de varias tareas o ejercicios.
2. Docente: usuario que se dedica a la enseñanza dentro de un curso.
3. Auxiliar: estudiantes que apoyan la labor docente en diversos ámbitos, ya sea en la corrección como dictando clases.
4. Ayudante: estudiantes que apoyan la labor docente con la corrección de evaluaciones.

3.4. Requerimientos funcionales

A continuación, se enumerarán los requisitos funcionales ideados para cada uno de los cuatro usuarios definidos. Estos requisitos basados en los en los roles se contemplan para el usuario que pertenece a ese rol dentro de un curso.

3.4.1. Estudiante

1. Puede acceder a un curso y visualizar secciones de grupo, tareas y eventos
2. Puede crear, unirse o salirse de un tipo de grupo
3. Puede indicar que está buscando unirse a un grupo
4. Puede ver las tareas disponibles
5. Puede responder grupalmente a una tarea (enviar una entrega)
6. Puede ver la sala que se le asigna a su grupo a una instancia de trabajo
7. Puede ver y responder a la retroalimentación realizada por un docente

Para los usuarios Docente y Auxiliar se considerarán los mismos casos de uso para simplificar los roles dentro de un curso:

3.4.2. Docente y Auxiliar

1. Puede ingresar a los alumnos de un curso utilizando su RUT
2. Puede ver los integrantes de un curso
3. Puede crear un nuevo tipo de grupo (Laboratorio, Proyecto, etc)
4. Puede ver los grupos formados para cada tipo de grupo
5. Puede crear una tarea para un tipo de grupo
6. Puede crear una nueva instancia de trabajo (con fecha y salas) para una entrega
7. Puede cargar a la aplicación la asistencia de los estudiantes a una instancia de trabajo
8. Puede evaluar y/o comentar una entrega realizada por un grupo

3.4.3. Ayudante

1. Puede ver las entregas que le corresponde revisar
2. Puede evaluar las entregas que le corresponde revisar
3. Puede ver la sala que se le asigna para asistir a una instancia de trabajo

3.5. Notificaciones

Colocar notificaciones en una aplicación de apoyo a la docencia puede mejorar la comunicación, la participación y la gestión de la clase, lo que beneficia tanto a docentes como a estudiantes al proporcionar una experiencia educativa más eficiente y efectiva.

Se plantea que cada usuario deba recibir notificaciones dependiendo del rol de el curso en el que está, esto para poder dar aviso de algún cambio en la aplicación que requiera una acción del usuario correspondiente, así agilizando e informando oportunamente.

Entregas por revisar

Notificación una vez al día, indicando el número de entregas pendientes por revisar. Esta notificación es recibida por los usuarios con el rol de Docente, Ayudante o Auxiliar.

Entregas asignadas para corregir

Notificación que se recibe cuando se cierra el plazo de una tarea. Cuando se cierra esta tarea, se realiza la división de entregas a los ayudantes.

Nueva corrección o comentario

Esta notificación es recibida por el estudiante cada vez que un docente o auxiliar le envía un comentario.

Nuevo evento de trabajo

Notificación que se envía a todos los estudiantes cuando se les asigna una sala para trabajar durante una instancia de trabajo.

Estas notificaciones podrían ser enviadas por correo o nativamente a través de la aplicación.

3.6. Diseño de la navegación de la aplicación

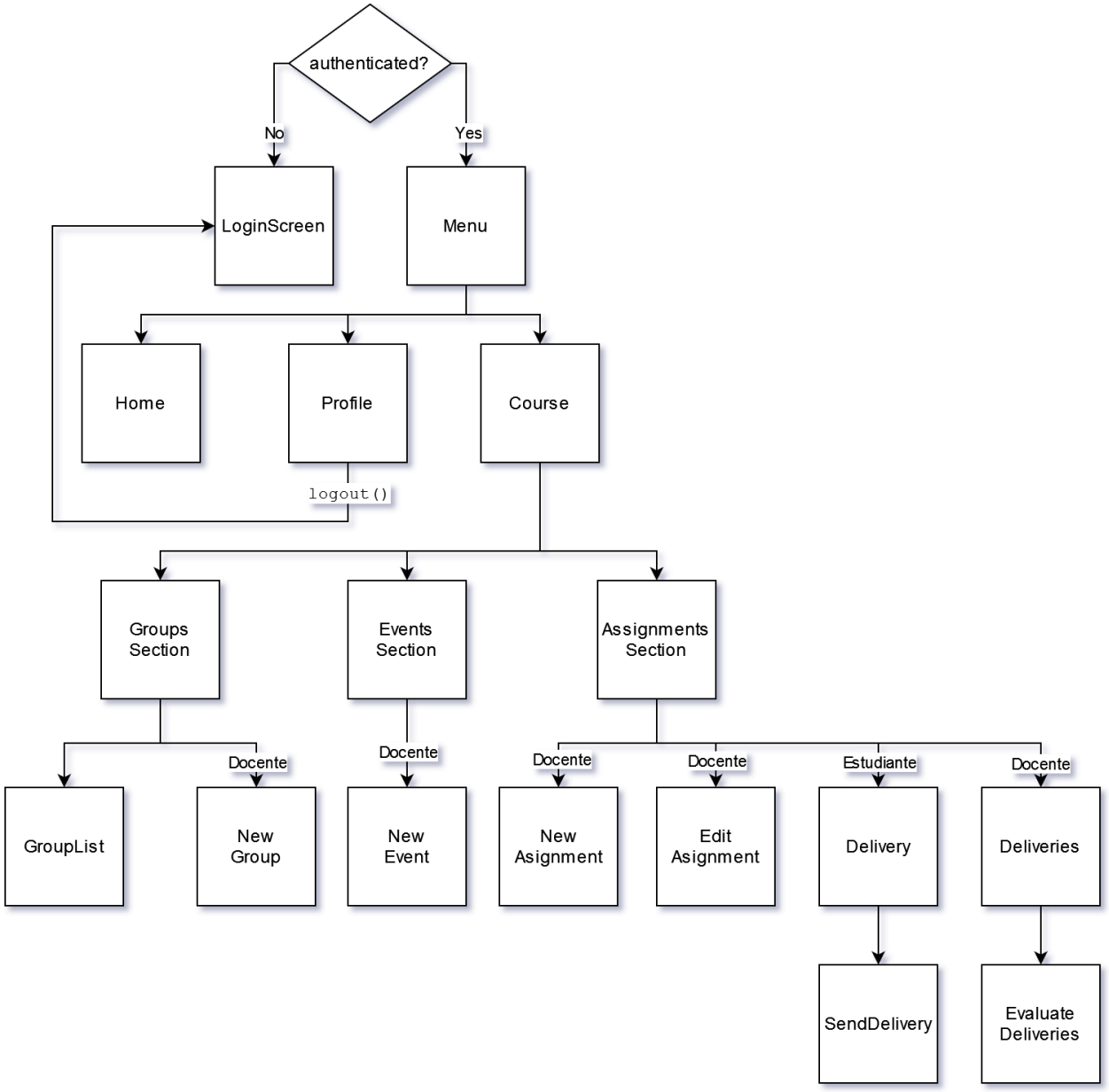


Figura 3.2: Diseño del flujo de navegación de la aplicación

A continuación, se presenta el diseño del flujo de navegación de la aplicación.

Este flujo presenta una primera bifurcación que permite mostrar una vista de ingreso si el usuario no se ha registrado aun. En caso de que se haya registrado, este puede ver el menú con opciones de navegar al Inicio (*Home*), el Perfil (*Profile*) o a cualquiera de los cursos de los que forma parte (*Course*).

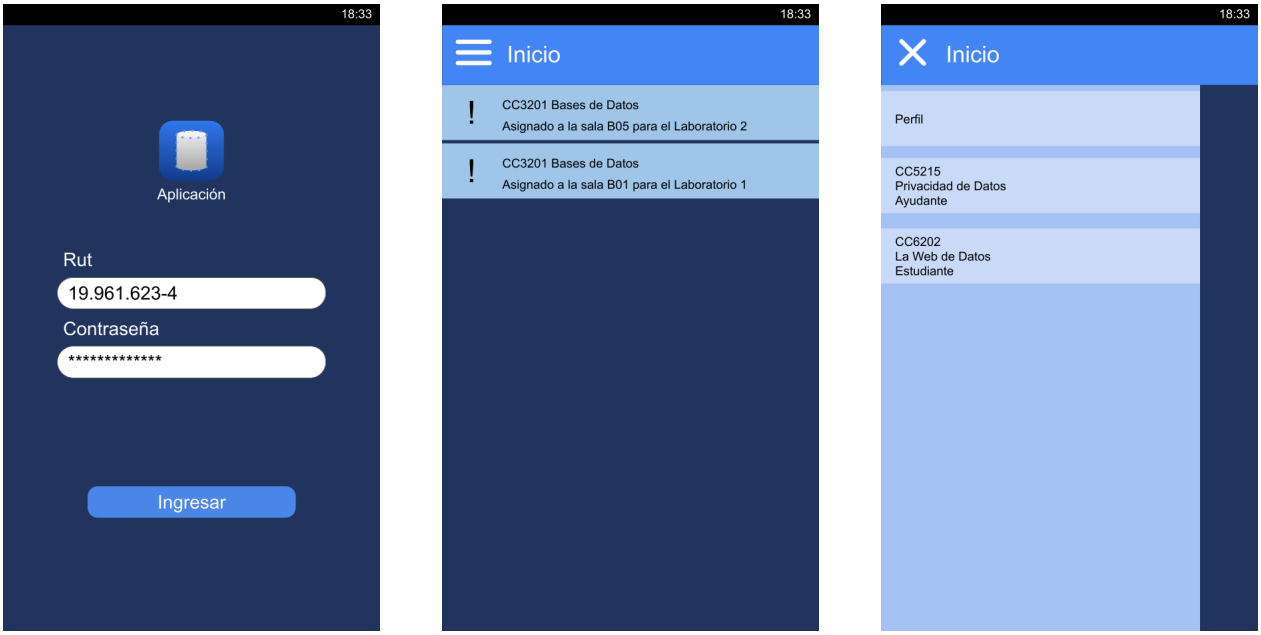
Una vez seleccionado alguno de sus cursos, podrá ver tanto la sección de grupos (*Groups*), eventos (*Events*) o tareas (*Assignments*).

En la quinta fila de la figura 3.2 se encontraran distintas vistas que contendrán la siguiente información:

- **GroupList**: corresponderá a la configuración de uno de los tipos de grupos seleccionado. En él, el estudiante podrá crear, unirse o salirse de un equipo (un equipo es un conjunto de estudiantes que participará en una misma entrega). Además podrá ver el resto de los equipos y ver quienes del resto de los integrantes no pertenece a un equipo.
- **NewGroup**: corresponderá a un formulario en el que un docente podrá crear un nuevo tipo de grupo (laboratorio, control, proyecto, etc.).
- **NewEvent**: corresponderá a un formulario en el que un docente podrá crear una nueva instancia de trabajo para una tarea en particular.
- **NewAssignment y EditAssignment**: corresponderá a un formulario en el que el docente podrá crear y editar, respectivamente, tarea, con un título, descripción y fecha de término.
- **Delivery**: corresponderá a una lista de todas las tareas del curso, ordenadas por fecha de entrega. En esta vista, los estudiantes podrán ver el estado de la ultima entrega de cada una de las tareas.
- **SendDelivery**: en esta vista, los usuarios, al seleccionar una tarea de la vista anterior, podrán ver más detalles de la tarea y realizar un envío a esta.
- **Deliveries**: docentes en esta vista lograrán ver, agrupadas por tarea, el número de entregas sin corregir y las corregidas.
- **EvaluateDeliveries**: en esta vista, los docentes, al seleccionar una tarea de la vista anterior, podrán ver una lista de las entregas realizadas y un menú desplegable para cada una de ellas para que sean evaluadas.

3.7. Diseño de interfaces

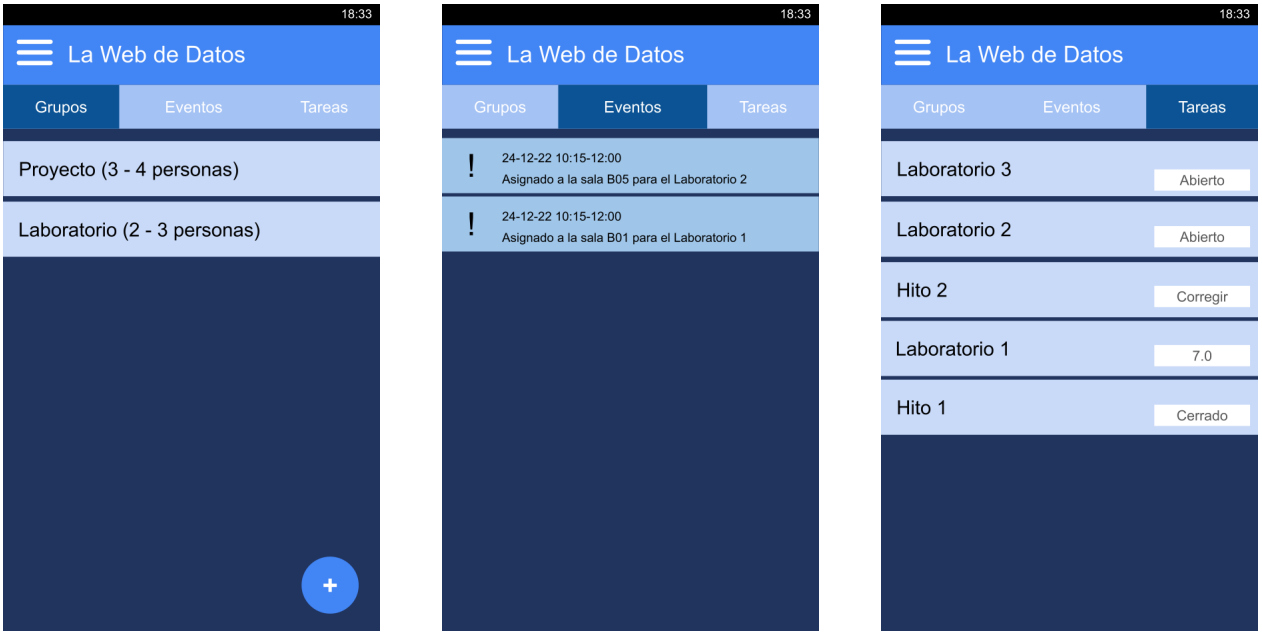
A continuación, se presentan las maquetas de las interfaces deseadas para la aplicación. Estas han sido diseñadas a partir de componentes que son comunes entre aplicaciones móviles y plataformas web ya que se desea que la aplicación sea intuitiva en el uso. Además, al utilizar componentes comunes permitirá un desarrollo más rápido de la aplicación, utilizando componentes ya creados por otros desarrolladores. Las maquetas presentadas solo en formato vertical.



(a) Mockup de ingreso (b) Mockup del inicio (c) Menú lateral

Figura 3.3: Mockups de ingreso, inicio y menú lateral.

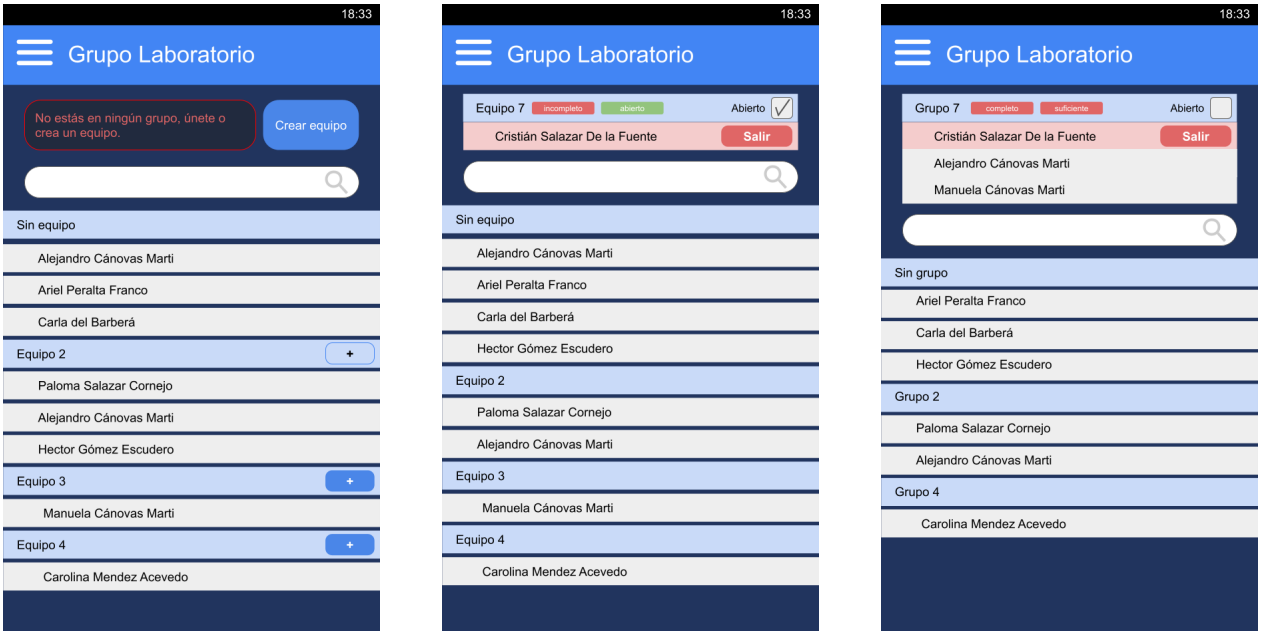
En la figura 3.3a se puede ver la vista de ingreso a la aplicación donde los usuarios ingresan sus credenciales para entrar. Una vez que ingresan, se les mostrará la pantalla de inicio (figura 3.3b), donde tendrán información relevante como las últimas notificaciones o informaciones importantes. Todas las vistas tendrán un título junto con un botón para acceder a un menú lateral desplegable (conocido como *drawer* en inglés o cajón), como se muestra en la figura 3.3c. En este menú desplegable podrán ver un botón por cada curso al que pertenezca y un botón que redirige a la vista del perfil, donde podrá ver toda su información personal.



(a) Mockup de grupos (b) Mockup de eventos (c) Mockup de tareas

Figura 3.4: Mockups de grupos, eventos y tareas dentro de un curso.

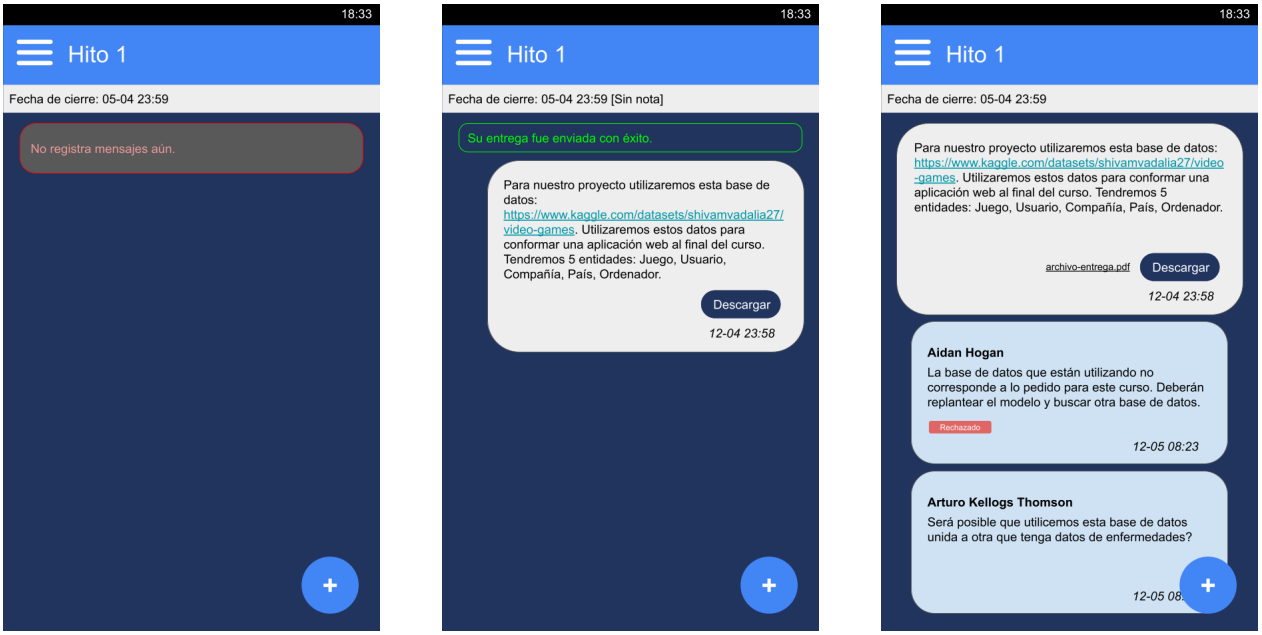
Al presionar sobre uno de los cursos del menú lateral, se redirigirá al usuario a la vista de un curso. En este podrá ver, dependiendo de la pestaña elegida, los grupos, eventos o tareas (figura 3.4). En la figura 3.4a se podrán ver listados los grupos disponibles para el curso, además de la cantidad permitida de estudiantes por equipo. Presionando en la pestaña "Eventos" se podrá acceder a la vista en la figura 3.4b, donde se podrán ver asignaciones de sala para los distintos eventos del curso. Por último, en la vista de "Tareas" de la figura 3.4c se verán las tareas listadas junto con un estado que sirva de orientación y resumen. Estas tres vistas tendrán un botón en la esquina inferior derecha, que redirigirá a un formulario para agregar un nuevo tipo de grupo, un nuevo evento o una nueva tarea, en caso de que el usuario sea un docente.



(a) Estudiante sin equipo (b) Equipo creado (c) Equipo completo

Figura 3.5: Mockups de distintos estados de un equipo del estudiante.

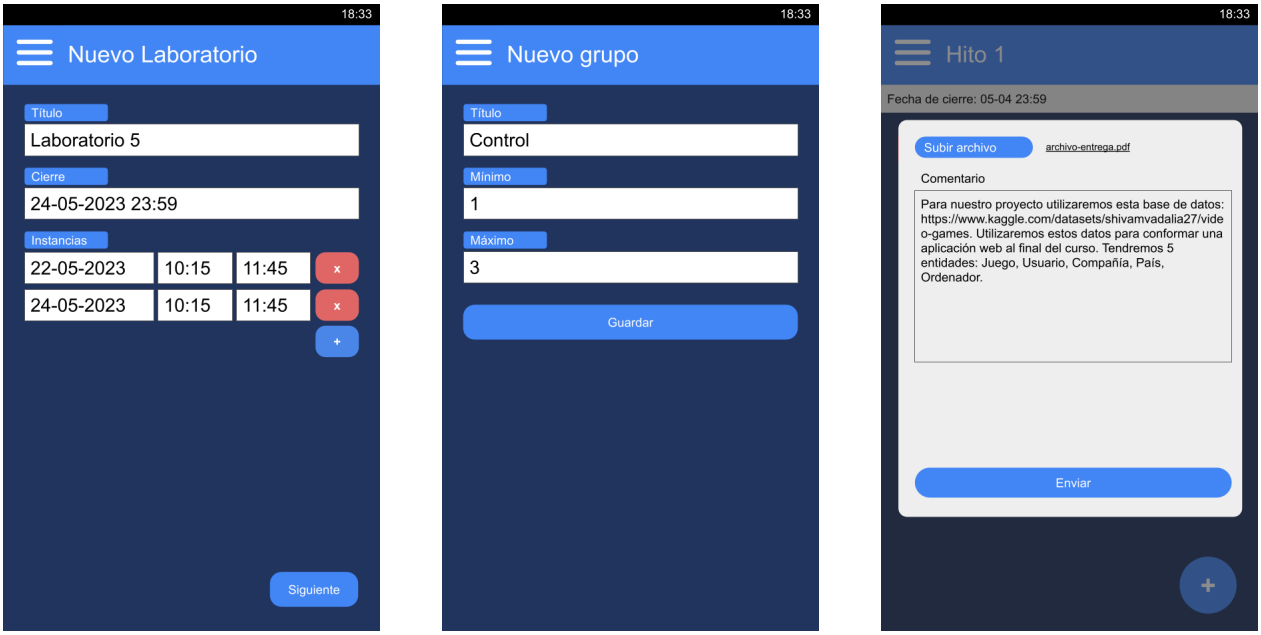
A través de los mockups de la figura 3.5 se explican como se dispondrá la información para distintos estados de un grupo de un estudiante. A esta vista se accede presionando sobre uno de los elementos de la lista de el mockup 3.4a. En la figura 3.5a se puede ver un mensaje que indica que el estudiante no tiene equipo, y al lado un botón para crear un nuevo grupo. También se le presenta una lista de los estudiantes agrupados por equipo. El estudiante tendrá botones que le permitirán unirse a los equipos que se encuentren abiertos y con el estado de suficiente o incompleto. En caso de que el equipo no se encuentre dentro de estos estados, el botón aparecerá como deshabilitado (como se ve en el equipo 2). En la figura 3.5b se verá que el usuario creó un equipo y se le mostrará la información del equipo al que pertenece en la parte superior de la pantalla, junto con un botón para salir del equipo. Si es que más estudiantes ingresan a un equipo, entonces se mostrarán agregados al equipo del usuario (figura 3.5c). El usuario que está marcado en rojo corresponde al que está autenticado. La tarjeta del equipo del estudiante tendrá una casilla de verificación que permitirá cerrar o abrir el equipo en caso de no querer que nadie más se una.



(a) Estudiante sin equipo. (b) Entrega enviada. (c) Entrega con comentarios

Figura 3.6: Mockups de distintos estados de la vista de entrega para una tarea.

A través de los mockups contenidos en la figura 3.6 se explica la lógica de entrega de una tarea y como se dispondrán los elementos de entrega y comentarios. A esta vista se accede presionando sobre una de las tareas listadas de la interfaz 3.4c. Se podrá ver la fecha de cierre independiente del estado de esta vista en la parte superior de la pantalla. La lógica de estas vistas es como de un chat con el equipo docente. En la figura 3.6a se muestra un mensaje indicando que no se registran entregas ni comentarios. Cuando un estudiante realiza una entrega, esta se muestra como un mensaje, con un comentario, un botón de descarga del archivo y la hora de entrega (figura 3.6b). Cuando un docente realiza comentarios sobre la entrega o un estudiante realiza otra entrega, estos se muestran también como mensajes (figura 3.6c).



(a) Formulario nuevo evento. (b) Formulario nuevo grupo. (c) Envío de tarea.

Figura 3.7: Mockups de algunos formularios de creación.

Por último, en la figura 3.7 se muestran algunos formularios de creación que tendrá la aplicación. El docente puede acceder a la figura 3.7a cuando presiona sobre el icono "+" de la vista de "Eventos" del curso. El docente puede acceder a la vista de la figura 3.7b cuando presiona sobre el icono "+" de la vista de "Grupos" del curso. Y a la vista 3.7c podrán acceder estudiantes cuando desean enviar una nueva tarea desde la vista de la figura 3.6.

3.8. Arquitectura de la aplicación

La aplicación desarrollada en esta memoria implementa un patrón de diseño de tres capas: Cliente o presentación, lógica del negocio y base de datos. En la figura 3.8 se muestra la arquitectura de la aplicación y como las diferentes capas se comunican entre ellas.

Bajo React Native tendremos tres componentes importantes:

- Navigation: refiere a la forma en que los usuarios se mueven dentro de la aplicación, navegando entre diferentes pantallas o vistas. Esta navegación puede ser gestionada mediante la navegación basada en pilas (*stack navigation*), la navegación de pestañas (*tab navigation*), la navegación de cajón (*drawer navigation*), entre otros. La navegación es fundamental para proporcionar una experiencia de usuario fluida y coherente dentro de la aplicación móvil.
- Components: son bloques de construcción fundamentales de la interfaz de usuario de la aplicación. Estos están diseñados específicamente para funcionar en dispositivos móviles, adaptandose al sistema operativo. Los componentes pueden ser simples, como botones o etiquetas de texto, o complejos, como listas desplegables o navegadores de

imágenes. Los componentes pueden ser creados por el desarrollador (componentes personalizados) o pueden ser proporcionados por bibliotecas de terceros.

- Service: en la aplicación se utilizará para permitir la comunicación con la API que contiene la lógica de la aplicación y maneja los datos (en otras palabras, comunicarse con el *backend*).

Bajo Djago Rest Framework se tienen cuatro módulos importantes:

- Views: son funciones o clases que procesan las solicitudes HTTP entrantes y devuelven respuestas. Estas manejan la lógica del negocio de la API, recuperando datos del modelo, serializando los datos y generando la respuesta HTTP.
- URL Patterns: los patrones de URL permiten enrutar solicitudes HTTP entrantes a las Views o vistas correspondientes. Esto permite estructurar y organizar la API, asignando endponints específicos a diferentes funciones o clases de vista.
- Serializers: son componentes que se utilizan para convertir objetos del modelo en formatos de datos compatibles con la web, como son JSON y XML y viseversa. Facilitan la conversión de los datos de los modelos de Django en respuestas HTTP comprensibles. permitiendo una comunicación eficiente entre la API y los clientes.
- Models: estas son clases que representan las estructuras de datos correspondientes a la aplicación. Cada uno de estos representan las tablas de la base de datos y definen tanto sus campos como las relaciones entre estas. Además, los modelos encapsulas la lógica de acceso y manipulación de datos de la aplicación. Finalmente, este módulo es el que interactúa con la base de datos a través de la capa de abstracción de Django ORM (Object-Relational Mapper).

Por último, se tiene una base de datos en PostgreSQL con la que interactua directamente Django REST Framework desde la capa de abstacción de Django ORM.

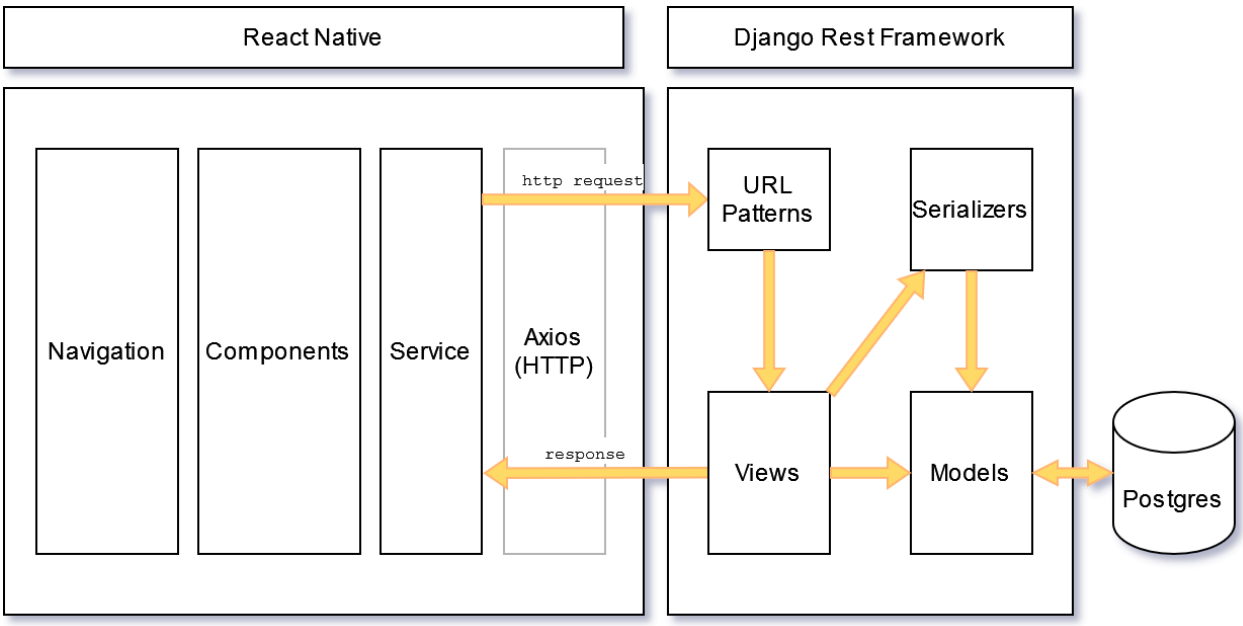


Figura 3.8: Arquitectura de la aplicación

Capítulo 4

Implementación

Para el desarrollo de la aplicación, acorde a la metodología elegida, se realizó un *backlog* con la división de tareas según importancia e interdependencia entre cada una de estas. Se describirá la aplicación implementada en el orden incremental en la que fue desarrollada. En primer lugar se trabajó sobre los usuarios y la forma que poseen para autenticarse. En segundo lugar, se trabajó sobre los cursos y los miembros de cada uno, abarcando roles de estos miembros dentro de un curso. Luego, se implementó todo lo correspondiente a la creación de tipos de grupos y la conformación de equipos de trabajo. Posteriormente, se implementaron funcionalidades de creación de nuevas tareas, entrega de desarrollos de las tareas, además de la creación de comentarios a cada una de las entregas y cambios de estado de las entregas. Por último, se implementó de forma incompleta, la creación de instancias de trabajo.

A continuación, se detalla y desarrolla sobre cada uno de estos puntos.

4.1. Usuarios y autenticación

Los usuarios que utilizan la plataforma, tanto estudiantes como el equipo docente, tienen una forma de autenticarse frente a la aplicación.

Usuarios

Como indica la documentación de Django, este framework incluye un sistema de autenticación de usuarios. Maneja cuentas de usuarios, grupos, permisos y sesiones de usuarios basados en *cookies*. Para esta memoria se utilizó la clase por defecto auth.models.User, sin realizarle extensiones o modificaciones. Esta clase contiene los siguientes atributos: *username*, que corresponderá al RUT del usuario, sin puntos ni guión. Luego tenemos el resto de los atributos: *password*, *email*, *first_name*, *last_name*. Estos atributos representarán la información inicial de todo usuario que utilice la aplicación.

UserProfile

Para almacenar posibles datos futuros del usuario como fotografía, descripción personal, número de teléfono, entre otros datos, se creó el modelo UserProfile, que contiene el campo *user*, que corresponde a un atributo OneToOneField[1] que permite la creación de una relación uno a uno, con la cual, desde el Usuario se puede acceder a su UserProfile y viceversa. Los demás atributos son *description*, *photo*, *email* y *phone_number*.

Autenticación basada en Tokens

Debido a la arquitectura de la aplicación, se utilizarán tokens para la autenticación y el control de acceso a los datos de la aplicación. El esquema de autenticación utilizado corresponde al implementado por Django REST Framework. Este es un esquema simple de autenticación HTTP basado en tokens. Este tipo de autenticación es adecuada para configuraciones cliente-servidor, como clientes nativos de escritorio y móviles.

Cada vez que un usuario nuevo ingresa a la aplicación por primera vez, se le asigna un token que no expira, con el que, al agregarlo a la solicitud HTTP, puede acceder a los demás datos que el servidor ofrece.

Vista de ingreso

Para hacer efectivo el ingreso de los usuarios a la aplicación, estos tienen una vista que les permite realizarlo, como muestra la figura 4.1a.

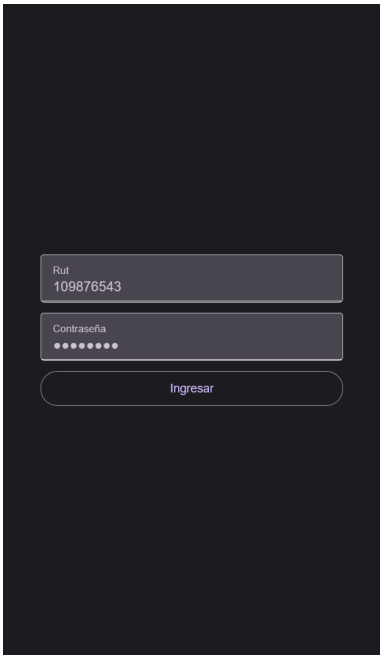
Inicio y vista del perfil

Una vez que el usuario ingresó sus credenciales correctamente, se le muestra la página de inicio en la que se muestra una descripción de la aplicación según se muestra en la figura 4.1b. Desde esta misma vista, el usuario puede acceder a un menú lateral o menú de cajón (figura 4.1c), desde la cual puede acceder a sus cursos o a visualizar su información personal y editarla, como indica la figura 4.2a.

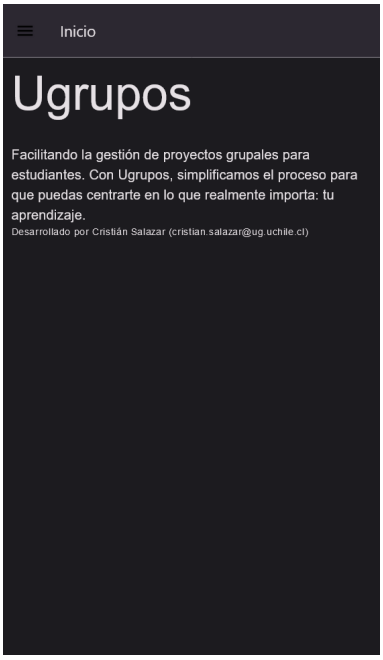
4.2. Cursos

Para que los usuarios puedan ver sus cursos y el rol que tienen dentro de estos, deben existir dentro del modelo de datos. A continuación, se detallará cada una de las clases implementadas en el modelo de datos. Cabe destacar que cada una de las clases descritas heredan de la clase models.Model de Django, lo que permite heredar métodos y atributos que no

[1] https://docs.djangoproject.com/en/4.2/topics/db/examples/one_to_one/



(a) Interfaz de ingreso



(b) Interfaz de inicio

(c) Menú lateral

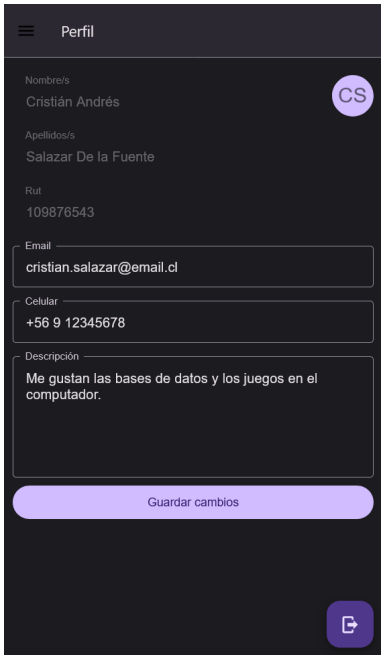
Figura 4.1: Interfaces de ingreso, inicio y menú lateral.

serán descritos, pero que son importantes. Entre ellos está el atributo *ID*, que corresponde a un entero que incrementa su valor automáticamente cuando se crea una nueva instancia del objeto y otros métodos para crear una nueva instancia en el modelo, editar o eliminarlo.

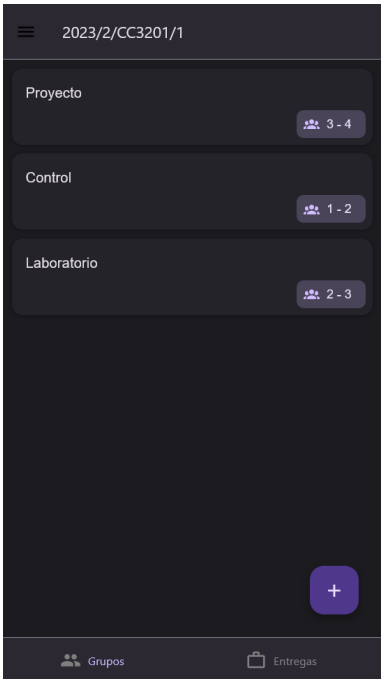
Course

Esta clase corresponde a la representación de un curso, el cual tiene los siguientes atributos:

1. *Code*, que corresponde al código del curso. Este código no cambia con el paso de los semestres y representa a un curso. Es de tipo texto.
2. *Name*, que corresponde al nombre en lenguaje natural que representa al curso, como lo puede ser Bases de Datos, Ingeniería de Software, entre otros.
3. *Year*, que indica el año en el que se dicta ese curso.
4. *Semester*, que indica el número del semestre en el que se dicta. Este es un número entero.
5. *Section*, que indica la sección de un curso, en el caso de que este se esté dictando dos o más veces en un mismo año y semestre.
6. *Active*, corresponde a un valor booleano para indicar si este curso sigue activo o no. En caso de no estar activo, esto indica que el curso no está siendo dictado en el momento.
7. *ID*, que sobreescribe la llave primaria heredada de la clase models.Model, para ser una composición de los atributos code, year, semester y section, unidos por un *slash* (/).



(a) Interfaz del perfil



(b) Interfaz de un curso

Figura 4.2: Interfaces de perfil y un curso.

Roles

Esta clase contiene solamente un atributo, *name* que indica el nombre de los posibles roles que puede tener un estudiante en un curso, los cuales son *Estudiante*, *Docente*, *Auxiliar* y *Ayudante*.

Members

Esta clase, como se mostró en la figura 3.1, corresponde a una relación de otras tres entidades: User, Role y Course. Esta representa a un miembro de un curso asociado a un rol dentro de este. Para modelar la relación, los tres atributos se definen como una llave foránea a las tres clases nombradas.

Interfaz de un curso

Como se puede ver en la figura 4.1c, en el menú lateral se pueden apreciar los cursos a los que pertenece el usuario junto con su rol. Al presionar uno de los cursos, el usuario es dirigido a la vista de un curso (figura 4.2b).

En la vista de la figura 4.2b, en la parte inferior, se aprecia una navegación de tipo pestaña (*tabs navigation* en inglés) en la cual se pueden ver dos botones que permiten acceder a las secciones de Grupos y Tareas, las cuales serán descritas en las siguientes dos secciones.

4.3. Grupos y equipos

Primero, se describirán los modelos de datos implementados para incluir las funcionalidades de crear tipos de grupo y unirse a un equipo, después se mostrarán las interfaces que le permiten al usuario realizar dichas acciones.

GroupType

Se tiene la clase GroupType para definir los tipos de grupos que pueden existir dentro de un curso (como Laboratorio, Proyecto, Control, entre otros). Estos tienen un nombre, cantidad de integrantes mínimo y máximo para considerar a un equipo enmarcado dentro de este tipo de grupo como válido, y una llave foránea a un curso al cual pertenece este tipo de grupo.

Team

A esta clase le corresponde representar un equipo, descrito como un conjunto de estudiantes, que trabajará en las tareas correspondientes a un tipo de grupo en particular. Esta clase tiene los siguientes atributos:

1. *name*, que corresponde al nombre personalizado que podrían querer asignarle los estudiantes pertenecientes al equipo.
2. *lfm* (por *looking for more*, en inglés), corresponde a un valor booleano para indicar que este equipo está en busca de más miembros.
3. *group_type*, es una llave foránea a algún tipo de grupo. Por ejemplo, este equipo podría pertenecer al tipo de grupo "Laboratorio".
4. *owner*, es una llave foránea al usuario que creó el equipo.
5. *members*, indica quienes son los miembros actuales del grupo. Este atributo es representado por Django con la clase ManyToManyField, que permite trabajar los integrantes como un conjunto de Members.
6. *state*, permite asignarle un estado a un equipo, que se explican en la siguiente sección.

Estados de un equipo

Como se mencionó anteriormente, un equipo puede tener un estado. Un equipo siempre tendrá alguno de estos, basados en los atributos de cantidad de integrantes mínimo y máximo del tipo de grupo:

1. **Incompleto**: Este es el estado correspondiente a un equipo que se encuentra con un número de integrantes menor que el mínimo declarado en el tipo de grupo al que pertenece. Cuando un equipo está en este estado, ninguna persona del equipo puede realizar una entrega para la tarea.
2. **Suficiente**: En este estado se encuentra un equipo cuando no está incompleto y el número de integrantes está bajo el máximo. En este equipo se pueden agregar más integrantes y pueden entregar.
3. **Lleno**: En este estado, el equipo no puede agregar más integrantes y pueden entregar.
4. **Exceso**: En este estado, el equipo no puede entregar ni agregar más integrantes, solo se pueden salir personas. Este estado nunca debería ocurrir, pero fue agregado para abarcar casos borde.

Para mantener el estado de equipo, se creo la clase TeamManager, que hereda de models.Manager con un método que agrega y otro que elimina un integrante a un equipo, actualizando el estado basado en máximos y mínimos de un tipo de grupo.

Como contexto, la clase Manager es una interface a través de la cual, las operaciones de consulta a la base de datos son proveídas a los modelos de Django. Al menos, por defecto, un Manager existe para cada uno de los modelos en la aplicación de Django[2]. En este caso, se crea este Manager para agregar o quitar miembro en un equipo, actualizando el estado de equipo.

Se tienen tres métodos que permiten cambiar el estado de un grupo al agregar o quitar un integrante:

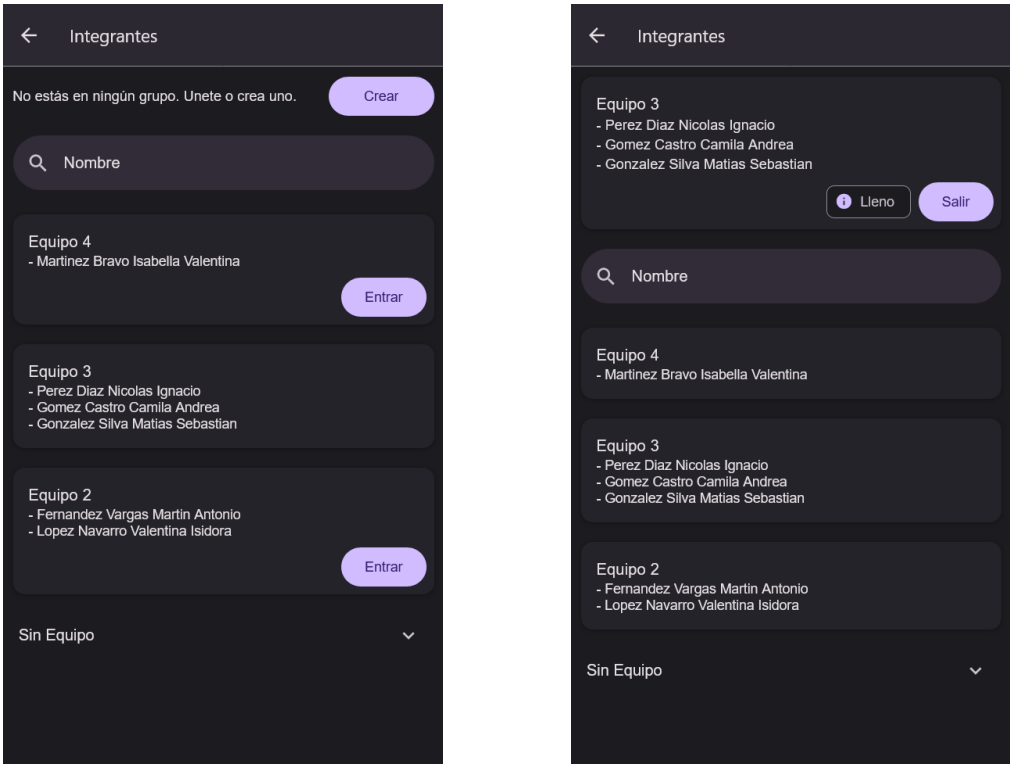
```
class TeamManager(models.Manager):
    def update_state(self, team):
        group_type = team.group_type
        members = team.members.count()
        if group_type.min_group_size <= members:
            if members < group_type.max_group_size:
                team.state = 1
            elif members == group_type.max_group_size:
                team.state = 2
            else:
                team.state = 3
        else:
            team.state = 0
        team.save()

    def add_member(self, team, member):
        team.members.add(member)
        self.update_state(team)

    def remove_member(self, team, member):
        team.members.remove(member)
        self.update_state(team)
```

Listing 4.1: Definición de TeamManager

[2]Información obtenida de https://docs.djangoproject.com/en/4.2/topics/db/managers



(a) Estudiante sin grupo (b) Estudiante con grupo lleno

Figura 4.3: Interfaces de formación de equipo.

En el listado de código 4.1 se aprecian los métodos add_member y remove_member que permiten agregar y quitar respectivamente miembros a un equipo, llamando al método update_state posteriormente, que actualiza el estado del grupo. El método update_state permite la actualización del estado basado en las directrices descritas al inicio de esta sección.

Interfaz de los tipos de grupos

En esta vista (figura 4.2b), se puede apreciar una lista con los distintos tipos de grupos que existen para ese curso. Además, en esta vista se puede ver cuál es la cantidad mínima y máxima de integrantes por grupo. Si el rol del miembro corresponde al de docente, este tiene un botón que le permite la creación de un nuevo tipo de grupo, indicando su nombre y el número de integrantes mínimo y máximo.

Interfaz de los equipos

Si se presiona sobre uno de los tipos de grupo, se podrá acceder a la vista que se muestra en la figura 4.3. En esta, se ven todos los equipos con sus integrantes hasta el momento y, en una lista desplegable al final, los integrantes sin equipo para ese tipo de grupo. También se presenta una barra de búsqueda que permite filtrar equipos en base a los nombres integrantes de cada una de ellas.

En caso de no tener equipo, la interfaz le señala al usuario que debe crear o unirse a uno (figura 4.3a). El usuario puede seleccionar alguno de los grupos disponibles que no estén llenos. Cada vez que un estudiante desea crear un nuevo equipo, se crea uno nuevo en la base de datos. Cuando un equipo se queda sin integrantes, este es eliminado. Una vez que se une, puede ver el estado de dicho grupo y optar por salirse (figura 4.3b).

El docente también puede acceder a esta vista, pero no se le muestran los elementos para unirse o crear un equipo.

4.4. Tareas

A continuación, se describirá el modelo implementado y las interfaces que permiten a estudiantes ver las tareas disponibles.

Assignments

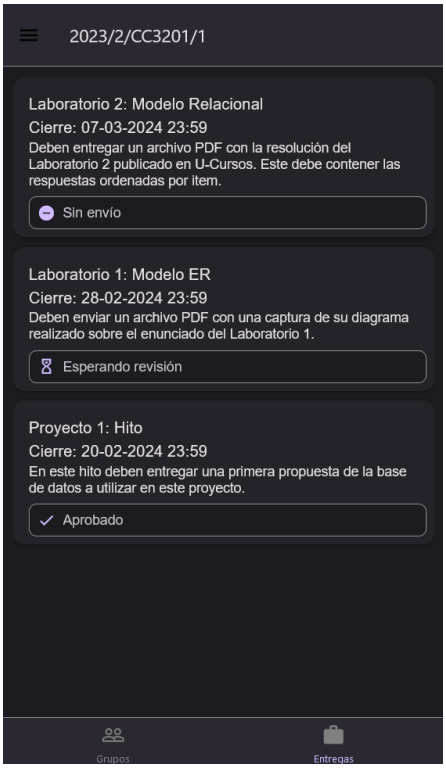
Con esta clase se representan las tareas que tiene un curso para un tipo de grupo en particular. Esta clase posee los siguientes atributos:

1. *group_type*, se indica con una llave foránea a qué tipo de grupo pertenece esta tarea.
2. *number*, corresponde a un numero entero que indica el número de la tarea dentro de este tipo de grupo. Por ejemplo, en estos titulos: Laboratorio 1, Proyecto 3, los números serían 1 y 3 respectivamente.
3. *description*, almacena una descripción o instrucciones adicionales de la tarea.
4. *title*, corresponde a un campo de texto opcional, que indica la unidad a la que corresponde la tarea. Un ejemplo podría ser: Modelo Entidad Relación.
5. *end_date*, con un tipo de dato de marca de tiempo (fecha y hora) indicando la fecha de cierre de la tarea, osea, cuando ya no se aceptan más envios de entrega.

Interfaz de las tareas

Como se puede apreciar en la figura 4.4a, el estudiante puede ver las tareas disponibles y publicadas por los docentes del curso. En esta vista se puede ver un estado de la última entrega realizada para esa tarea (se desarrolla más sobre esto en la siguiente sección) y la fecha de término. Estas tareas están ordenadas por la fecha de cierre de forma descendente.

Si el usuario corresponde a un docente dentro del curso, este podrá ver el número de entregas recibidas en cada una de las tareas (figura 4.4a). Las tareas recibidas para cada tarea se calculan como todas las entregas que tienen un estado distinto al de “inválida”. Al igual que en la interfaz de los tipos de grupo, el docente tiene un botón que le permite



(a) Vista de tareas (estudiante).

(b) Vista de tareas (docente).

Figura 4.4: Lista de tareas de estudiantes y docente.

crear una nueva tarea, indicando el nombre, número, una descripción, al tipo de grupo que pertenece esta tarea y la fecha límite de entrega.

4.5. Entregas

Delivery

Dado que estudiantes pueden enviar la resolución de sus tareas, registra los envío de estas realizadas por miembros de equipos para alguna tarea en particular. Esta clase contiene los siguientes atributos que contienen información de la entrega:

1. *timestamp*, indica la fecha y hora en la que fue enviada la tarea. Esta siempre será menor que el tiempo de término de la tarea correspondiente.
2. *file*, correponde a un campo de tipo FileField, que permite manejar y almacenar direcciones de archivos en la ruta configurada para tales efectos. Este campo será el archivo enviado por el miembro del equipo.
3. *description*, indica de forma opcional una pequeña observación de la tarea ingresada por el miembro que envía la resolución de la tarea.

4. *assignment*, una llave foránea a la tarea a la que corresponde la entrega.
5. *delivered_by_user*, indica qué usuario envíó la tarea. Corresponde a una llave foránea a User.
6. *delivered_by_team*, indica qué equipo envió la tarea. Corresponde a una llave foránea a Team.
7. *delivered_by_members*, indica, utilizando un ManyToManyField, qué miembros fueron parte de la entrega. Esto es una redundancia con respecto a Team, pero se están copiando los miembros del Team correspondiente a este campo, con el objetivo de que queden registrados como integrantes de una entrega aunque el equipo sea modificado en el futuro.

Los últimos tres datos permitirán llevar un registro exhaustivo sobre la creación y pertenencia a los grupos.

Además, contiene información sobre quién evaluó o evaluará la entrega:

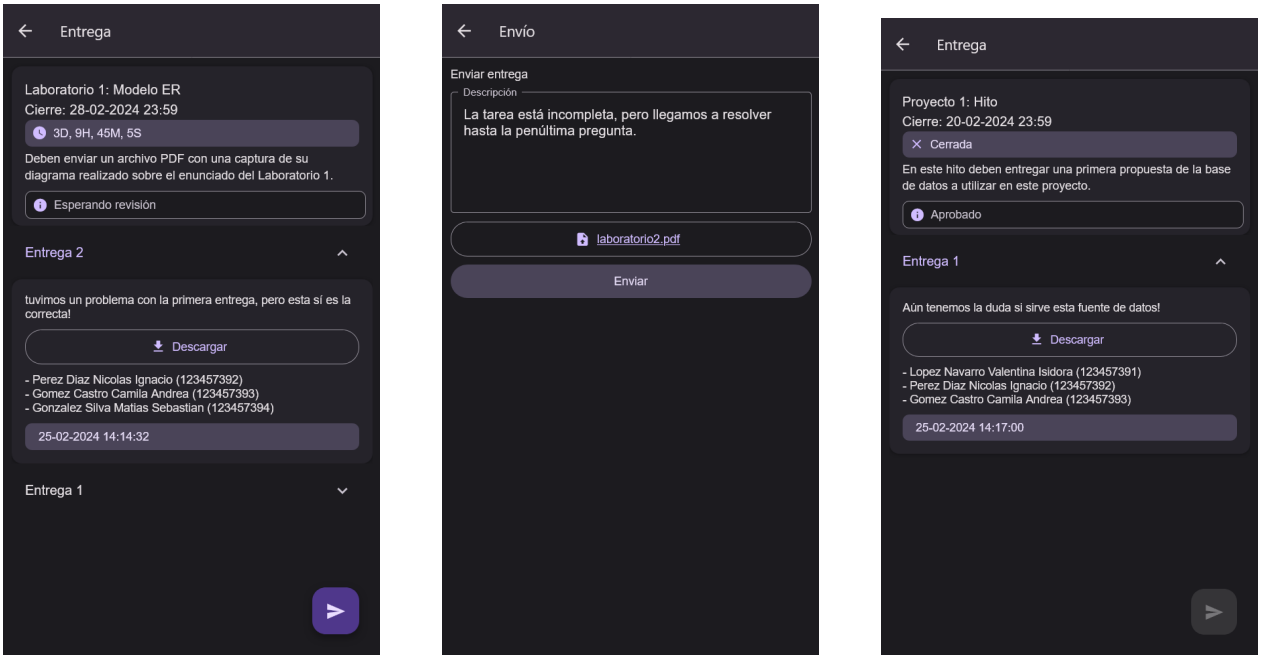
1. *grade*, que corresponde a un número flotante que puede ser nulo, indicando la nota obtenida en esta entrega luego de ser evaluada. Esta solo puede ser modificada por una persona del equipo docente del curso.
2. *state*, indica el estado de una entrega. Estos estados serán descritos más adelante.
3. *evaluator*, indica, en caso necesario, quienes de los miembros, con una llave foránea, es el o la responsable de evaluar la resolución de esta entrega.

Estado de una entrega

Las entregas (Delivery) pueden tener alguno de los siguientes estados:

1. **Esperando corrección**: Son las entregas pendientes a revisar por parte del equipo docente.
2. **Aprobado**: Significa que esta entrega ha sido evaluada y aprobada por sobre la nota reprobatoria o es suficiente para continuar en el curso.
3. **Reprobado**: Significa que esta entrega fue evaluada con una nota inferior al límite reprobatorio o que no ha sido suficiente para cumplir con los objetivos de la tarea.
4. **Corregir**: Esta entrega fue evaluada y necesita de algunas modificaciones, basadas en alguna retroalimentación, para ser aprobada.
5. **Inválida**: Corresponde a las entregas que ya no son válidas por no ser la última entrega para esta tarea.

Estos estados están implementados en el modelo de datos a partir de una lista de tuplas que asignan un número a cada uno de estos estados, desde el 0 al 4.



(a) Entregas de una tarea (b) Formulario de entrega (c) Tarea aprobada y cerrada

Figura 4.5: Interfaces de las tareas de estudiantes.

Entregas inválidas

Una entrega será clasificada como “inválida” cuando exista alguna otra entrega más reciente de algún miembro de dicha tarea. Cada vez que se envía una nueva entrega, todas las entregas anteriores de los miembros de esta nueva entrega quedan con el estado de “inválida”, sin importar el estado que tengan.

Vista de una tarea

Los usuarios acceden a esta vista al presionar sobre una de las tareas que se muestran en la lista. Cuando estudiantes acceden a esta vista, pueden ver el nombre de la tarea, la fecha de cierre, una cuenta regresiva a la fecha de cierre, la descripción de la tarea y el estado que posee la última entrega de la tarea. También, estudiantes pueden ver todas las entregas que existan con su usuario dentro de los miembros, independiente de su estado actual. Si no ha enviado una tarea, muestra un mensaje indicando que no se han realizado entregas (figura 4.5a).

Además, tienen un botón en la esquina inferior derecha que les permite enviar una tarea a través de un formulario. En esta se muestra un campo de texto para colocar alguna observación sobre la tarea a entregar y un botón para seleccionar un archivo desde el dispositivo (figura 4.5b).

En el caso de que un estudiante no cuente con equipo en estado “suficiente” o “lleno” para el tipo de grupo de la tarea seleccionada, este no podrá acceder al formulario de envío

de tarea, apareciendo un mensaje que indica que no forma parte de un grupo válido, en vez del botón de la esquina inferior derecha.

En caso de que la tarea esté cerrada, el usuario tampoco podrá acceder al formulario de envío de tarea (figura 4.5c).

4.6. Comentarios

En esta sección se describirá la entidad Comment implementada para permitir el almacenamiento de mensajes del equipo docente a los estudiantes, sobre una de las entregas realizadas, para entregar una retroalimentación de forma ordenada. Luego, se explicarán las vistas que tiene el docente para realizar retroalimentación a las entregas de los estudiantes. Por último, se revisará la vista que poseen los estudiantes para revisar los comentarios recibidos en una entrega en particular.

Comment

Los atributos que tiene esta clase son: *timestamp*, correspondiente a la marca de tiempo del mensaje enviado; *comment*, que indica el texto del comentario; *delivery*, una llave foránea a la entrega sobre la que se está comentando y *author*, que es una llave foránea al usuario que realizó el comentario.

Vista de revisión de entregas

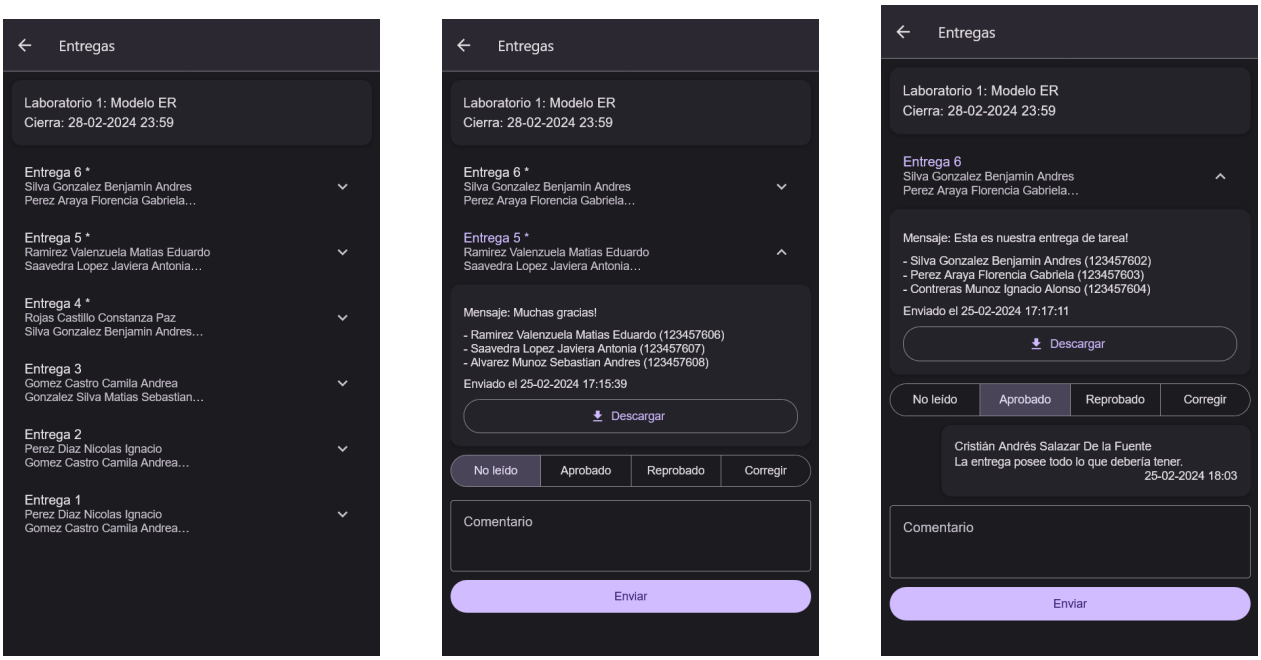
Cuando un docente presiona sobre una de las tareas, este puede ver todas las entregas recibidas hasta el momento (figura 4.6a). En esta vista puede ver una lista con elementos desplegables. Cada elemento corresponde a una entrega. Los nombres de entrega que tienen un asterisco al final corresponden a las que tienen el estado de "Esperando corrección", con el objetivo de indicar cuales aun no han sido revisados.

Al presionar en uno de los elementos, se despliega información sobre los integrantes de la entrega, fecha de entrega, un botón para descargar la entrega, botones segmentados para cambiar el estado de la entrega y un campo para incorporar observaciones sobre la entrega (figura 4.6b).

Cuando el docente envía comentario a la entrega, esta se agrega a la lista de comentarios (figura 4.6c).

Vista de comentarios recibidos

En la figura 4.7 se puede ver que el estudiante recibió un comentario de un docente.



(a) Lista de entregas (b) Detalle de entrega (c) Comentario de un docente

Figura 4.6: Vistas de las entregas (docente).

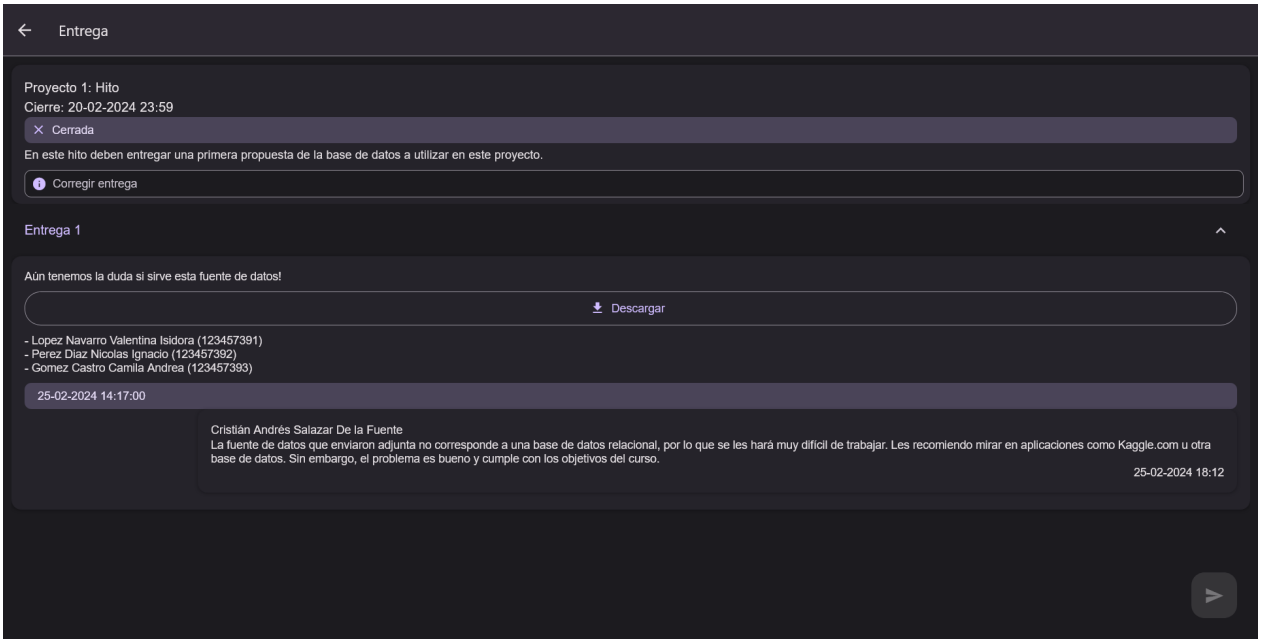


Figura 4.7: Comentario recibido por docente (vista en aplicación web).

4.7. Instancias de trabajo

Por último, se describen las clases del modelo de datos implementadas restantes, que corresponden a la modelación de las instancias de trabajo para la distribución de equipos en las salas disponibles. Si bien fue implementado el modelo, que será descrito a continuación, no se implementaron las funcionalidades que permiten distribuir de forma balanceada a estudiantes y equipo docente a las respectivas salas.

Classroom

Esta clase permite la representación de las salas disponibles para ser usadas para las instancias de trabajo. Para permitir la asignación de salas de forma equilibrada se tiene la capacidad de las salas y su nombre.

WorkInstance

Con esta clase, se representan las instancias de trabajo o eventos en los que se trabaja síncronamente en una de las tareas del curso. Esta clase posee cuatro atributos:

1. *assignment*, correponde a una llave foránea al Assignment (tarea) en la que se va a trabajar.
2. *classrooms*, indica las salas asignadas para esta instancia de trabajo. Este es un campo de tipo ManyToManyField, lo que permite trabajar las salas como un conjunto de salas.
3. *start_date* y *end_date*, corresponden a la fecha y hora de inicio y término, respectivamente, de la instancia de trabajo.

AssignedClassroom

Indica qué equipo completo fue asignado a alguna de las salas para trabajar sobre la instancia de trabajo. Por lo tanto, esta tabla contiene una llave foránea al equipo, a la instancia de trabajo y la sala en la que se trabajará.

Capítulo 5

Validación

5.1. Diseño experimental

Con la finalidad de evaluar y validar que la aplicación desarrollada en esta memoria es usable y util, se diseñó el siguiente análisis cualitativo, en donde se evalúa la usabilidad y el potencial para agilizar el curso. Este cuestionario contempla 11 preguntas, 10 son en torno al diseño de la aplicación, para responder en forma de alternativas y la última pregunta corresponde a una de respuesta libre, en la que, opcionalmente, se pueden dejar comentarios sobre la aplicación en general.

Para llevar a cabo el experimento, se requería que fueran estudiantes actuales de alguno de los cursos con metodologías de trabajo y entregas grupales o estudiante que hayan cursado este tipo de cursos.

En este caso, para la realización del experimento, 18 estudiantes del curso CC3201 Bases de Datos del semestre 2023-2, durante el horario de laboratorio y con sus grupos de ese momento, se ofrecieron para probar la aplicación y realizar su entrega a través de ella. Debido a que este experimento ocurrió en una etapa avanzada del curso, no fue necesario explicar las reglas del curso.

La realización de los experimentos ocurrió dentro de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, en la misma sala de clases en las que se realizan. Los estudiantes no fueron observados durante la realización y tenían permitido conversar con sus compañeros de equipo.

El experimento consistía en seguir una lista de instrucciones sin recibir ayuda o apoyo en el término de la tarea:

1. Ingrese a ugrupos.dcc.uchile.cl
2. Ingrese en la aplicación utilizando su rut.
3. Ingrese al curso actual (CC3201 Bases de Datos).

4. Forme un grupo con sus compañeros de laboratorio.
5. Entreguen un archivo que corresponda al desarrollo de su laboratorio 8.
6. Esperar el comentario del docente sobre su entrega y efectuar la acción que le pide el docente.

Con respecto al último punto, los estudiantes recibían un comentario de un docente que en ese momento era el memorista quien otorgaba comentarios a las entregas. Este comentario era el siguiente:

> Evaluar aplicación en el siguiente enlace: https://forms.gle/4BzerHEv3ZxViGbA6

Las preguntas utilizadas para el análisis en torno a la experiencia previa se explicarán en la siguiente sección.

Al finalizar el experimento se le explicó y la evaluación a las y los estudiantes el objetivo tanto del experimento como de la aplicación.

5.2. Evaluación de usabilidad

A continuación se detalla el diseño de la evaluación de usabilidad, sus resultados y el analisis posterior.

5.2.1. Diseño

La evaluación de usabilidad es de real importancia para medir la experiencia del usuario de una forma rápida, a tiempo. Una mala experiencia de usuario, para el caso específico de esta memoria, implica el fracaso del objetivo principal, ya que queremos simplificar un flujo que actualmente es lento.

Se diseñó un cuestionario en el que las preguntas contenidas están dentro de la lista de principios que debe seguir una interfaz de usuario respecto a su diseño según Jakob Nielsen[1].

1. Creo que me gustaría usar la plataforma frecuentemente.
2. Pienso que el sistema es innecesariamente complejo.
3. El sistema es fácil de usar.
4. Creo que necesitaré asistencia de un técnico para usar el sistema.
5. Pienso que las funcionalidades del software están bien integradas.

[1]Lista de principios que debe seguir una interfaz de usuario respecto a su diseño según fueron publicados en 1995 por Jakob Nielsen.

6. La plataforma tiene demasiadas inconsistencias.
7. Creo que la mayoría de las personas aprenderán a usar el sistema rápidamente.
8. El sistema no es agradable para su uso.
9. Me siento seguro al usar el sistema.
10. Necesito aprender muchísimas cosas antes de lograr avanzar en el uso del software.
11. Escriba a continuación comentarios generales sobre la aplicación. Se puede referir tanto a la interfaz como al propósito de la aplicación.

Para cada una de estas afirmaciones el usuario debe seleccionar que tan de acuerdo o en desacuerdo se encuentra con cada una de ellas. El usuario presenta las siguientes alternativas de respuesta:

1. Totalmente de acuerdo
2. De acuerdo
3. Neutral
4. En desacuerdo
5. Totalmente en desacuerdo

Dichas respuestas representan un puntaje que va del 1 al 5, en el orden presentada en la lista anterior.

Para la última pregunta, la respuesta correspondía a un texto de respuesta larga.

5.2.2. Resultados

A continuación, se presentarán los resultados de las preguntas 1 a la 10, mostrando la ocurrencia de las respuestas entregadas para cada una de las alternativas de las respectivas preguntas. Estas respuestas se presentan en dos gráficos distintos, uno con las respuestas a las preguntas impares (figura 5.1) y otra de las respuestas pares (figura 5.2), ya que las preguntas impares y pares corresponden a aspectos positivos y negativos respectivamente, sobre la usabilidad de la aplicación.

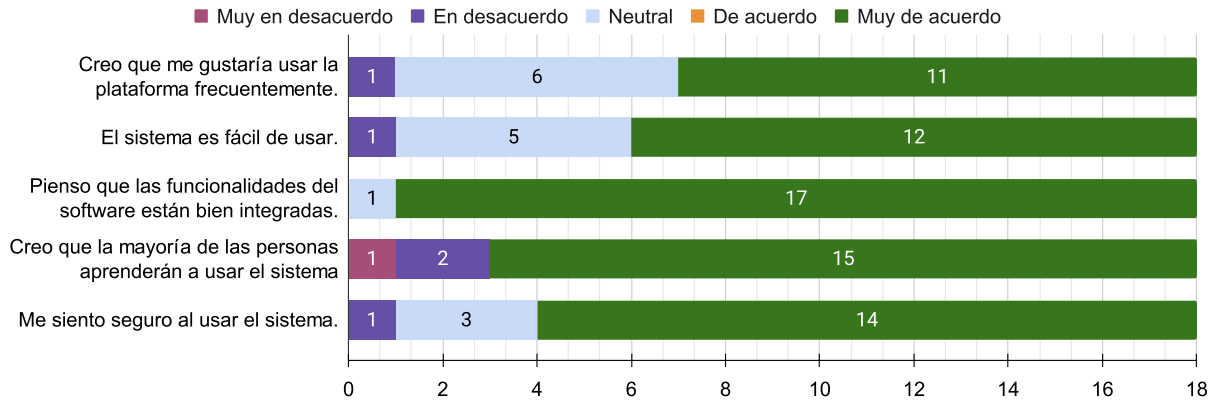


Figura 5.1: Respuestas preguntas impares.

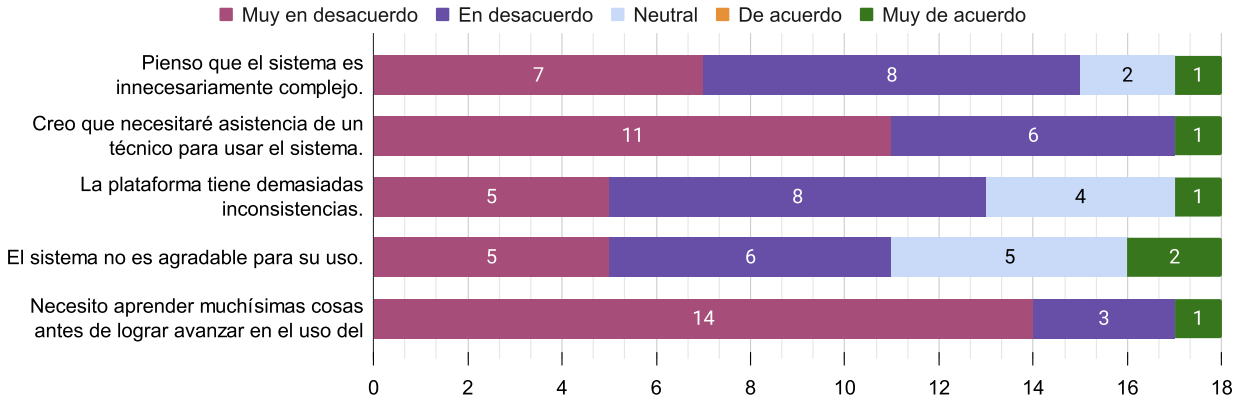


Figura 5.2: Respuestas preguntas pares.

A continuación se presenta un resumen de los comentarios expresados por quienes probaron la aplicación:

- Creo que quizás sea buena idea de que al refrescar la pagina, no se se cargue en el inicio necesariamente, sino que donde yo refresco la pagina, por ejemplo, cuando estoy creando un grupo y quiero ver el feedback, tengo que cargar la pagina y esto hace que vuelva al inicio y tenga que volver a la sección donde estaba, esto haría que fuera mas dinámica la pagina. A su vez, los botones aveces confunden un poco, como que cuesta aveces ver bien en donde hay que apretar, pero creo que eso puede ser mas algo de CSS mas que otra cosa.
- Es buena idea que cualquier integrante del grupo pueda sobre escribir la entrega, para que de esa forma se entregue 1 sola forma y no varias para facilitar la corrección de los ayudantes, esto lo probé y funciona así, dejando historial de entregas previas también lo cual me gusta. Me gusta tener los archivos de versiones pasadas disponibles junto a su feedback.
- Para entrar a laboratorios hacer clic sobre el icono de 2-3 integrantes no funciona, hay que hacer clic en otra parte.

- Agregar función que se pueda arrastrar los archivos al interfaz para entregar.
- Los botones están un poco escondidos.
- El botón de entregas está en un lugar raro, me tomó un segundo ubicarlo.
- No sé si es algo implementable, pero al estar en, por ejemplo, entregas de documento y al recargar página ésta vuelve al inicio, teniendo que volver a esa parte de la pestaña y siendo esto algo tedioso (y estaría bueno tener un tema claro).
- No sé si por temas de seguridad está bien que den los ruts de todos. Hay otros temas que son pocos intuitivos de ver, como el botón de enviar entrega, también debiese de haber una opción para tener que aceptar a los integrantes, no me gustaría que se meta una persona aleatoria a mi grupo.
- Lo que más me complicó al usar la página fue que los elementos clickeables no se diferenciaban de los no clickeables. Otro detalle fue que me costó encontrar el botón para enviar una entrega (al estar en la esquina contraria al texto), pero a eso es fácil acostumbrarse. El resto de la página me pareció sencilla de utilizar.
- Tuve dificultades al momento de revisar mis entregas. No había ningún botón claro, y tuve que presionar sobre la palabra "Tareas"(que no es un botón, o sea, el cursor no cambió su formita a çlickear"), y ahí recién pude ver mis entregas.
- Me gustó la aplicación, cumple con su funcionalidad.
- Las funcionalidades están buenas, pero el diseño de la pagina podría mejorar para hacer la experiencia más agradable.
- Intenté arrastrar la imagen del Fiu a la ventana para cargarlo y en su lugar la ventana fue reemplazada por la imagen del Fiu. No supe cómo expandir la tarea, esperaba que el botón de diferente color realizara el efecto, pero en su lugar hubo que pinchar la palabra. Sólo me cargó un tema purpura oscuro (quizá enforced por el sistema, no lo sé), sería agradable ser introducido inicialmente con un tema claro que mostrara un buen contraste entre elementos y me permitiera aprender la ubicación de los botones (los cuadros de diálogo y los botones lucían idénticos on first approach); luego permitirme opcionalmente utilizar un tema oscuro. Me gusta el tema oscuro, pero mi falta de experiencia con la UI se volvió un obstáculo en recibir bien el layout.
- Buen software, soluciona el problema de entregar por grupos así que podría implementarse en otros cursos.

5.2.3. Análisis

En la figura 5.1 se puede ver la mediana calculada para cada una de las preguntas, en base a la transformación de resultados cualitativos a cuantitativos, y el puntaje final calculado para todo el experimento. Este puntaje final del experimento fue calculado en base a la siguiente fórmula. Se tendrán dos expresiones: preguntas positivas y pregunas negativas (PP y PN respectivamente). Las siguientes ecuaciones transforman los resultados a una escala

que va desde el 1 al 100, donde si el puntaje es mayor a 68, el resultado de la evaluación es considerado como positivo, según lo que establece la escala SUS.

$$PP = P_1 + P_3 + P_5 + P_7 + P_9 - 5 \tag{5.1}$$

$$PN = 25 - (P_2 + P_4 + P_6 + P_8 + P_{10}) \tag{5.2}$$

$$Puntaje = (PP + PN) \cdot 2{,}5 \tag{5.3}$$

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**Puntaje**
Promedio	4,2	1,9	4,3	1,6	4,9	2,1	4,4	2,3	4,5	1,4	**82,5**

Tabla 5.1: Mediana de cada pregunta y el puntaje final de la encuesta.

Por lo tanto, basado en el puntaje calculado, se puede ver que el resultado de la evaluación es considerado como positivo.

Analizando más a fondo los resultados, interesa revisar cuales fueron las preguntas con peores resultados. Esto significa que se obtuvieron respuestas menos favorables para la usabilidad y más alejadas del máximo puntaje (que corresponde a la media de 5 para las preguntas positivas y 1 para las preguntas negativas).

Pregunta	**Puntaje**	**Distancia**
El sistema no es agradable para su uso.	2,3	1,3
La plataforma tiene demasiadas inconsistencias.	2,1	1,1
Pienso que el sistema es innecesariamente complejo.	1,9	0,9
Creo que me gustaría usar la plataforma frecuentemente.	4,2	0,8
El sistema es fácil de usar.	4,3	0,7
Creo que necesitaré asistencia de un técnico para usar el sistema.	1,6	0,6
Creo que la mayoría de las personas aprenderán a usar el sistema rápidamente.	4,4	0,6
Me siento seguro al usar el sistema.	4,5	0,5
Necesito aprender muchísimas cosas antes de lograr avanzar en el uso del software.	1,4	0,4
Pienso que las funcionalidades del software están bien integradas.	4,9	0,1

Tabla 5.2: Mediana del puntaje de cada pregunta, ordenado por la distancia a la mediana máxima.

Podemos ver en la tabla 5.2, se puede ver que las preguntas con distancia mayor a 1 son las que indican que el sistema no es agradable para su uso y que presenta inconsistencias. Estas disconformidades se pueden asociar a los comentarios expresados en la pregunta 11, donde los comentarios hacen referencia a no poder arrastrar archivos a la aplicación, elementos que parecen botones presionables, áreas que esperan que podrían ser presionables pero no lo son, un botón en un lugar poco esperado y que al recargar la página, se redirige a la pantalla de inicio, cuando se espera que queden en la misma vista.

Con respecto a la pregunta 11, 14 de las 18 personas dejaron comentarios sobre la aplicación. En estas 14 respuestas:

- 10 personas comentan sobre no poder encontrar en primer instancia lo que estaban buscando, que elementos que esperaban fueran cliqueables no lo fueran o que intentaron arrastrar un archivo a la aplicación y no lo lograron.
- 2 comentan sobre que la navegación no funciona como esperaban, ya que al refrescar la página, se les redirige a la pantalla de inicio.
- 1 corresponden a ideas de nuevas funcionalidades
- 4 corresponden a mensajes de felicitaciones por la aplicación.

Con esto se puede ver que la mayoría de los comentarios nombran aspectos de la usabilidad, consistencia y navegación en la aplicación, por lo que se debe tomar en cuenta estos comentarios para mejorar la experiencia de usuario en futuras iteraciones del proyecto.

Capítulo 6

Conclusión

6.1. Conclusión del trabajo

A continuación, se expondrán las conclusiones del trabajo con respecto al cumplimiento de los objetivos inicialmente planteados, luego se realizarán algunas reflexiones sobre trabajo de título expuesto y, por ultimo, se desarrollarán algunas posibilidades para el trabajo futuro de este proyecto.

6.1.1. Objetivos específicos

A continuación, se muestran los objetivos específicos planteados para el presente trabajo y de qué forma fueron abordados.

6.1.2. Objetivos específicos

1. **Reemplazar el flujo actual de conformación de equipos de un curso**: La aplicación fue desarrollada para poder reemplazar el flujo actual de conformación de equipos de forma exitosa. Además, este flujo pudo ser evaluado por usuarios reales en la evaluación de usabilidad, entregando resultados positivos.
2. **Implementar la recepción de entregas de las tareas en formato PDF o archivo de texto por parte de un equipo**: Como fue descrito anteriormente, la aplicación soporta entregas de todo tipo de archivos a una tarea.
3. **Integrar a la aplicación el registro de la asistencia desde U-Cursos para relacionar entregas con asistencia.**: Este objetivo no pudo ser implementado.
4. **Permitir la distribución de equipos de estudiantes a las salas de trabajo**: Este objetivo no pudo ser implementado.

5. **Implementar la asignación automática de ayudantes para corregir cada entrega**: Nuevamente, esta es una funcionalidad que no pudo ser diseñada ni implementada.

6. **Implementar notificaciones que permitan dar aviso de cambios en la aplicación que requieran una acción del usuario**: Si bien, fueron diseñadas qué notificaciones deberían recibirse y con qué periodicidad, esta funcionalidad no logró ser implementada.

6.1.3. Objetivo general

El objetivo general de este proyecto fue descrito como el siguiente:

"Implementar una aplicación multiplataforma (móvil y web) que permita asistir y apoyar a docentes y estudiantes en la gestión de cursos masivos con metodologías grupales."

Según los objetivos específicos y el desarrollo de esta memoria, se puede llegar a la conclusión de que el objetivo general se cumple, ya que, aunque no se lograron desarrollar todos los aspectos planificados, este sistema si permite el apoyo a la docencia y la gestión de cursos masivos.

6.2. Reflexión

En primer lugar, y con respecto a la experiencia personal, la creación desde cero una aplicación que permitiera mejorar un flujo existente, fue una experiencia positiva y llena de aprendizajes, en el sentido de que el memorista experimentó el desarrollo de un proyecto de software, desde la concepción de la idea hasta la implementación y evaluación de esta. Además, al ejecutar las pruebas de evaluación de usabilidad, logró probar la importancia que tiene esta etapa en los procesos de implementación y que permite dilucidar nuevas posibilidades para el sistema a desarrollar.

En segundo lugar, al hablar de la aplicación de los conocimientos aprendidos durante la carrera de computación, el memorista logró aplicar tanto habilidades blandas como duras. Por un lado, se debió conversar con quienes usarían la aplicación en un futuro, junto con transmitir ideas de forma clara para recibir opiniones y apreciaciones, dentro de la fase de diseño y concepción de la solución. Por otro lado, conocimientos tanto en redes como de programación lograron ser aplicadas en la implementación de esta memoria.

En tercer lugar, las tecnologías utilizadas en esta memoria fueron las correctas por dos razones. La primera, porque el memorista se encontraba familiarizado con estas tecnologías, lo que permitió una implementación más expedita, disminuyendo los tiempos de aprendizaje. La segunda, porque estas tecnologías, en ningún momento, presentaron una limitación funcional que no permitiera la continuación del proceso de desarrollo.

En cuarto lugar, al reflexionar sobre el nivel de desafío de esta memoria, se presentaron al-

gunos problemas que fueron resueltos. Uno de ellos fue pensar en cómo considerar una entrega como invalida de forma algorítmica, cuando estas reglas en la práctica siempre fueron transmitidas en palabras. Por esta razón, se puede concluir que este trabajo fue suficientemente desafiante para el memorista.

En quinto lugar, con respecto al diseño de las vistas, este fue un proceso de observación de las interfaces más conocidas para el general de las personas, con tal de tener una aplicación con una distribución de elementos que fuese familiar e intuitiva para los usuarios finales.

En sexto lugar, referente a la estimación del tiempo y la carga de trabajo, esta fue subestimada, confiando en el ímpetu del memorista de querer hacer una aplicación completa y que abarcara todos los problemas existentes, lo que se refleja en el incumplimiento de todos los objetivos planteados. Sin embargo, se logró ajustar y priorizar las tareas que agregaban más valor a la aplicación, con el fin de concluir esta memoria con un producto mínimo que abarcara los problemas más grandes.

Por último, mencionar que aunque no se cumplieron todos los objetivos planteados ni se implementó la aplicación completa deseada, esta trabajo si corresponde a una primera iteración de lo que puede ser un sistema de apoyo a la docencia y gestión de alumnos en cursos masivos con metodologías grupales.

6.3. Trabajo futuro

En un principio, el trabajo a futuro a realizar sobre la aplicación sería la implementación de la evaluación no solo en base a estados, si no que a notas, para que luego estas notas puedan ser exportadas a un formato soportado por U-Cursos. Luego de que se implemente esta funcionalidad, realizar una evaluación con respecto al tiempo invertido en la corrección por parte del equipo docente, si este es menor o mayor al utilizar la aplicación, versus los métodos actuales.

Es importante desarrollar las notificaciones, ya que estas permitirán recordar tanto a estudiantes como docentes sobre cuales son sus siguientes pasos a realizar para agilizar la gestión, disminuyendo los tiempos de entrega de retroalimentación.

Además, sería importante aplicar los cambios mencionados en los comentarios durante la evaluación de usabilidad como persistencia en la navegación luego de refrescar la página, indistinción entre elementos cliqueables y no cliqueables, interfaz adaptada para el navegador (ya que esta siempre fue diseñada de forma vertical), entre otros comentarios.

Se podría evaluar la publicación de esta aplicación en tiendas de aplicaciones móviles como App Store para IOS o Play Store para dispositivos Android, sin embargo, se cree que esto no es necesario si es que se tiene una versión web accesible desde el navegador desde cualquier dispositivo, además de que implica la inversión en recursos para su publicación en estas tiendas.

También se podría evaluar la implementación de las demás funcionalidades, en el caso de que las evaluaciones sobre tiempo invertido sean favorables para esta solución, con el objetivo

de seguir agregando funcionalidades que permitan agilizar la gestión docente en los demás aspectos a mejorar.

Por último, se podría pensar en presentar este trabajo, sus avances y resultados, a los desarrolladores de la aplicación U-Cursos, para que evalúen la posibilidad de desarrollar funcionalidades similares que estén integradas a la actual aplicación de la Universidad de Chile.

Bibliografía

[1] Expo. "https://docs.expo.dev/core-concepts". [Online; Accedido el 25-06-2023].

[2] Flutter. "https://flutter.dev/". [Online; Accedido el 02-07-2023].

[3] Ionic. "https://ionicframework.com/". [Online; Accedido el 25-06-2023].

[4] React Native. "https://reactnative.dev/". [Online; Accedido el 02-07-2023].

[5] System Usability Scale. "https://measuringu.com/sus/". [Online; Accedido el 26-09-2022].

[6] Technology Acceptance Model. https://acceptancelab.com/technology-acceptance-model-tam. [Online; Accedido el 26-09-2022].

[7] Universidad de Chile. U-Cursos. "https://www.uchile.cl/presentacion/vicerrectoria-de-asuntos-economicos-y-gestion-institucional/convenio-de-desempeno/sistemas-de-gestion/u-cursos". [Online; Accedido el 26-03-2023].

[8] Google. Hojas de Cálculo. "https://workspace.google.com/products/sheets/". [Online; Accedido el 26-03-2023].

[9] Universidad Adolfo Ibáñez. Presentación WebC. "https://www.youtube.com/watch?v=zSkFSPUVeIA". [Online; Accedido el 26-03-2023].

[10] Gehrke J. Ramakrishnan, R. *Sistemas de gestión de bases de datos.* McGraw-Hill Interamericana de España S.L., 2006.